Ensemble Learning for Extremely Imbalced Data Flows

Jordan Frery





N° d'ordre NNT : 2019LYSES034

École Doctorale ED488 Sciences, Ingénierie, Santé

Ensemble Learning for Extremely Imbalanced Data Flows

Apprentissage Ensembliste sur des flux de données extrêmement déséquilibrés

Thèse préparée par **Jordan Frery**
au sein de l'**Université Jean Monnet de Saint-Étienne**
en collaboration avec **Atos Worldline**
pour obtenir le grade de :

Docteur de l'Université de Lyon
Spécialité : **Informatique**

Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d Optique Graduate School,
Laboratoire Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France.

Table of Contents

List of Figures

List of Tables

Introduction

Machine learning is the study of designing algorithms that learn from training data to achieve a specific task. The resulting *model* is then used to predict over new (unseen) data points without any outside help. This data can be of many forms such as images (matrix of pixels), signals (sounds,...), transactions (age, amount, merchant,...), logs (time, alerts, ...). Datasets may be defined to address a specific task such as object recognition, voice identification, anomaly detection, etc. In these tasks, the knowledge of the expected outputs encourages a *supervised learning* approach where every single observed data is assigned to a label that defines what the model predictions should be. For example, in object recognition, an image could be associated with the label "car" which suggests that the learning algorithm has to learn that a car is contained in this picture, somewhere. This is in contrast with *unsupervised learning* where the task at hand does not have explicit labels. For example, one popular topic in unsupervised learning is to discover underlying structures contained in visual data (images) such as geometric forms of objects, lines, depth, before learning a specific task. This kind of learning is obviously much harder as there might be potentially an infinite number of concepts to grasp in the data. In this manuscript, we focus on a specific scenario of the supervised learning setting: 1) the label of interest is under represented (e.g. anomalies) and 2) the dataset increases with time as we receive data from real-life events (e.g. credit card transactions). In fact, these settings are very common in the industrial domain in which this thesis takes place.

Problems and Motivations

Today, IT companies are much more involved in AI research than a few years ago. Lately, we saw the creation of new research centers such as Deepmind in 2010, Google Brain in 2011 (Alphabet), FAIR (Facebook AI Research) in 2013 and OpenAI in 2015. In Figure 1, we present the number of publications from

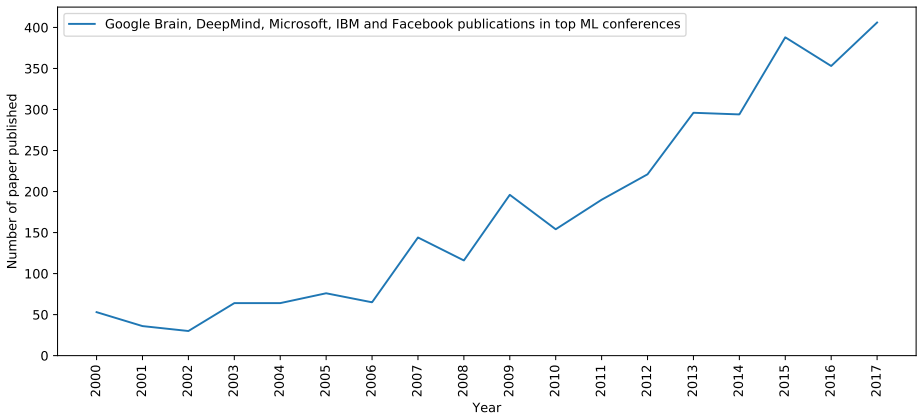

Figure 1: Number of publications per year by the 5 most advanced companies in Artificial Intelligence for top-tier conferences in machine learning.

such companies throughout the years [1] in prestigious machine learning conferences and a recent craze for companies to publish and share their most recent research is noticeable. This trend is spreading to many companies for two main reasons: the development of **hardware** and the **data** available. Indeed, the hardware improved a lot over the past decades with 1 TFLOPS (Tera FLoating-point Operations Per Second) costing around 1 million euros in 2000 dropping to 30 euros in 2017. Naturally, companies started to invest in building more powerful computing infrastructure to handle their massive amount of data. This latter is what makes a huge difference between the academia working mainly on public datasets and the industry having terabytes of data. In addition, they also gather the biggest computing infrastructure in the world, making them prevalent actors for the use of machine learning. Note that most of machine learning algorithms used today such as neural networks, boosting, random forest, logistic regression, SVM, and many more were created around the 2000's but the accessibility to more computing power and more data pushed the limits of these algorithms forward.

Nevertheless, these algorithms have been developed in confined settings where datasets are clean and relatively small. These new datasets coming from the industry offer new challenges. In this manuscript, we tackle several important points raised by their nature. It is important to note that real datasets coming from the industry is a rare commodity in the public domain. There are many different reasons why this data are kept away from the public research such as privacy and the

[1] Note that data were collected using the Google Scholar search engine thus there might be some omitted papers.

fact that they are an asset for the company owning them. However, recently, many companies started publishing extracts of their data fully anonymized through a competition format (e.g. kaggle) which makes them worthless from a business point of view but priceless for the data science community. That being said, it is clear that gathering a lot of data is costly in terms of human work especially because of the labelling part (for supervised learning) where one has to assign a ground truth to every data point. Moreover, data are more and more subject to many different social constraints. The most common one is the data privacy which can partially be solved by anonymizing the data at the potential cost of a loss of information available that could be relevant for a learning algorithm. As of today, the General Data Protection Regulation (GDPR) European laws have made the use of data legal only under very strict rules which makes the use of AI models more complex. In addition, these datasets, built over real-life events, are subject to many processes that often introduce some noise (e.g. human mistakes on labelling).

In this manuscript, one important focus is made on imbalanced datasets where the class of interest (e.g. fraudulent transactions) is under represented. At Atos Worldline, the company where this thesis has been done in collaboration with the Hubert Curien laboratory, we witness this issue in extreme and unprecedented cases where the fraudulent transactions appear about once out of two thousand times. We will show that in such a case, the evaluation metric is primordial and that the state of the art on imbalanced supervised learning suffers from biases induced by public datasets that are often very small.

Today, many companies use machine learning models in production. However, they are often being obsolete in the short term due to different concept drifts through time. For example, spam detection, fraud detection, anomaly detection, recommendation systems or click predictions are constantly evolving problems (e.g. new fraudulent strategies, new anomalies, new users' taste, ...) and models must adapt quickly to the changes in patterns. In production, for the majority of companies using machine learning, the role of a data scientist is often to maintain the models by watching the performances through metrics and retraining the models from scratch when needed. This is certainly not the optimal scenario. We would rather like a model that could adapt itself automatically as the data arrive and learn through time. Indeed, in real-life applications, data does not come in a finite set but rather arrive in a stream that never stops, defining actual events in the real-life. This thesis also takes a step forward solving this problematic.

Context of this thesis

Context of this thesis This thesis is part of a collaboration between the academy and the industry. *Worldline* is a company focused on e-payment services and has a special role to do with e-payment security. Indeed, several banks rely on their fraud detection system in order to anticipate on fraudulent behaviours and block fraudsters from stealing too much money before the card holders discover the fraud. Obviously, Worldline has access to different information for every transaction made. However, today, the fraud detection system mainly relies on fraud experts who build the so-called *expert rules* defined after analysis and investigations on the transactions. This is costly and in the long term, unrealistic. This is where machine learning comes in. Substantial amounts of data arrive every day in Worldline's system with multiple kinds of information. This thesis aims at adapting existing machine learning techniques to the challenges that offers the fraud detection problem. More generally, we tackle the supervised anomaly detection problem with two main constraints: the class imbalance problem and the continuous data feed.

Contributions and Structure of the Manuscript

This manuscript contains two main contributions and is structured in four different chapters. In Chapter 1, we introduce machine learning fundamentals used throughout this manuscript and present the general class imbalance domain with its imbalance learning methods and its evaluation metrics. We finally present some of the most famous ensemble methods with a focus on boosting that we use throughout this thesis.

Chapter 2 presents the specific application case in which this thesis took place: credit card fraud detection. We present different methods and show that, in our specific case, they introduce many constraints and biases that are complicated to handle with machine learning models. A large experimental study is made on a private dataset from Worldline to highlight the previous point. In these experiments we show some drawbacks behind different well-known performance metrics in the class imbalance case. We further conclude that metrics independent of this threshold better estimates the potential performance of a model. This brings us to the first main contribution.

Optimization of the average precision

Chapter 3 is our first contribution where we study the supervised anomaly detection problem. We propose an approach based on a learning to rank strategy by

optimizing different smooth surrogate of the Average Precision (AP), a particularly suited metric in the context of class imbalance data, in a Stochastic Gradient Boosting algorithm. We show that using AP is much better to optimize the top rank alerts than other commonly used measures. This learning to rank approach fits in the machine learning context where we wish to assist to the day-to-day job of human experts. This contribution was followed by a patent on the credit card fraud detection application.

Online Non-Linear Boosting

In the previous contribution, we mainly worked with the standard gradient boosting algorithm that uses linear combination. This latter naturally averages the performance of the models in the combination. It turns out that we could take advantage of non-linear combination to exploit the full potential of the models in the combination. In Chapter 4, our second main contribution, we study how we can make such combinations and take into account another important point of these real-life applications: the continuous flow of data. This contribution lies in the online learning domain where models must learn "on the fly" as examples arrive. We propose a new online boosting algorithm that uses more advanced combinations than in standard linear gradient boosting. We end this manuscript by a general conclusion, open questions and perspectives.

Table 1: Notations.

Notation	Description
R	Set of real numbers
$\mathcal{X}$,$\mathcal{Y}$	Input Space, Output Space
x	Vector
d	Number of dimensions in x
y	Target ground truth of example x
F_T	Ensemble model with T different classifiers
h_t	t^{th} weak learner in a boosting model
$f(\cdot)$	Function
$\mathbb{E}(\cdot)$	Expectation
$R_{true}(\cdot)$	True risk
$R(\cdot)$	Empirical risk
τ	Decision threshold
ρ	Imbalance ratio

Chapter 1

Preliminaries

Abstract

In this chapter we introduce several notions used throughout this manuscript. We formally define the supervised learning setting, the ensemble methods and more specifically, boosting. Lastly, we present learning methods and metrics of the state of the art for imbalanced datasets.

1.1 Supervised Machine Learning

In this section, we define precisely the setup for a supervised machine learning problem. In this type of learning, as for humans, the algorithm learns from observations and gets a feedback known as the ground truth. We first define a sample:

$$x \in \mathcal{X} \subseteq \mathbb{R}^d,$$

where $\mathcal{X}$ is the input space typically defined over $\mathbb{R}^d$ with d being the number of dimensions/features of a vector x such that we have $x = \{x^1, x^2, ..., x^d\}$. In this framework we also have the target y of the example x:

$$y \in \mathcal{Y} \subseteq \mathbb{R},$$

with $\mathcal{Y}$, the output space, discrete or continuous over $\mathbb{R}$. In this manuscript, we mainly focus on binary classification where $\mathcal{Y} = \{-1, 1\}$.

In practice, we have a training set $\mathcal{S}$ of size M defined as $\mathcal{S} = \{x_i, y_i\}_{i=1}^{M}$ where the M instances are supposedly independently and identically distributed (i.i.d.) according to an unknown joint distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.

Now that we have established the basic notations, we can describe formally what supervised learning means. In this framework, we generally want to find a function $f \in \mathcal{F}$ where $\mathcal{F}$ is the function space that maps the input features $\mathcal{X}$ to the target output space, $\mathcal{Y}$,

$$f \in \mathcal{F} \rightarrow \mathcal{Y},$$

where f is the function that predicts y given x for any (x, y) drawn from $\mathcal{D}$. In other words, we want to find $f(x)$ the function that best approximates $F(x)$, the true (unknown) function of the problem at hand. However, the real world has a lot of noise induced by missing features, wrong labelling, etc... We define ϵ the irreducible error that we are not able to recover from such that $y = F(x) + \epsilon$ (this ϵ also relates to the Bayes error which is the error of the Bayes optimal classifier).

In order to find the best function f for a given problem, we first need a performance metric. Let us define the loss function $\ell(\cdot, \cdot)$ that takes both the predicted output of the model $f(x)$ and the expected label y. As we later present, this loss function can be of many forms but it generally focuses in evaluating the agreement between $f(x)$ and y. We first define the notion of *Generalization Error* (or True Risk) $R_{true}(\cdot)$ which is the expected error of our model f over $\mathcal{D}$:

$$R_{true}(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}(\ell(f(x), y)).$$

In practice, we are only given a restricted training set $\mathcal{S}$ where every data point is assumed to be drawn randomly from the distribution $\mathcal{D}$ and that every example is generated independently from the others. This is the most common assumption made in machine learning which state that the data is independently and identically distributed (*i.i.d.* assumption). Thus, generally, the access to the expected error over $\mathcal{D}$ is impossible and we rather compute its empirical counterpart R using $\mathcal{S}$:

$$R(f) = \frac{1}{M} \sum_{i=1}^{M} \ell(f(x_i), y_i).$$

The empirical risk is proven to converge to the true risk (Hoeffding, 1963) $R(f) \xrightarrow{M\rightarrow\infty} R_{true}(f)$. In the end, we are looking for f such that $f = \inf_{f\in\mathcal{F}} R(f)$. A straight forward approach would be to minimize the empirical risk to find the function f such that:

$$f = \inf_{f\in\mathcal{F}} R(f) = \inf_{f\in\mathcal{F}} \frac{1}{M} \sum_{i=1}^{M} \ell(f(x_i), y_i).$$

However, optimizing over all possible $f \in \mathcal{F}$ functions may end up with a function that would perfectly fit the samples from the training set with a high generalization error. This phenomenon is known as *overfitting* and is explained by the following uniform convergence PAC bounds (Valiant, 1984) (or generalization error) derived from the *Hoeffding's inequality* (Hoeffding, 1963)

$$R_{true}(f) \leq R(f) + \sqrt{\frac{\log(|\mathcal{F}|) + \log(\frac{2}{\gamma})}{2M}}, \tag{1.1}$$

where $|\mathcal{F}|$ defines the number of functions in the search space $\mathcal{F}$ and $1 - \gamma$ is the probability for Eq. 1.1 to hold. This equation states that the generalization error becomes bigger as $|\mathcal{F}| \to \infty$ and tends to decrease as $M \to \infty$. Moreover, if the space of function $\mathcal{F}$ is large then finding the right model becomes computationally unrealistic. Note that in the infinite case (*i.e.* $|\mathcal{F}| = \infty$, *e.g.* when $\mathcal{F}$ is the family of hyperplanes in $\mathbb{R}^d$), we need to resort to a complexity measure to estimate the expressiveness of $\mathcal{F}$. An example of such measure is the VC (Vapnik Chervonenkis) dimension (Vapnik, 1971).

In machine learning, a famous notion is the bias-variance trade-off. The bias represents the average prediction error of the model $f(x)$ on the true function $F(x)$. It is defined as follows:

$$\mathbf{Bias}(f(x)) = \mathbb{E}\left(f(x) - F(x)\right).$$

A high bias tends to mean that the model is too simple leading often to a true risk relatively high. This phenomenon is known as *underfitting* where the model does not learn enough on the training set. The variance, on the contrary, represents the variability of the model with regard to the data.

$$\mathbf{Var}(f(x)) = \mathbb{E}\left(f(x)^2\right) - \mathbb{E}\left(f(x)\right)^2.$$

In practice, the simpler the model (*i.e.* the smaller the number of parameters to learn) the smaller the variance. As we previously mentioned, the true function that we want to approximate is given through a set of observations that are subject to noise. The risk of having a model with high variation is to induce a model that tries to approximate this noise. The intuition of this notion in Figure 1.1 where overfitting (high variance) and underfitting (high bias) are represented in function of the model complexity. In summary, having too complex models makes them prone to overfitting while too simple models are not able to learn the idiosyncrasies of the target concept. In practice we can estimate the curve of the true risk by evaluating the model on a test set $\{x_i, y_i\}_{i=1}^{T} \in \mathcal{S}_{test} \sim \mathcal{D} \backslash \mathcal{S}$. The optimal trade-off is the one that minimizes both bias and variance.

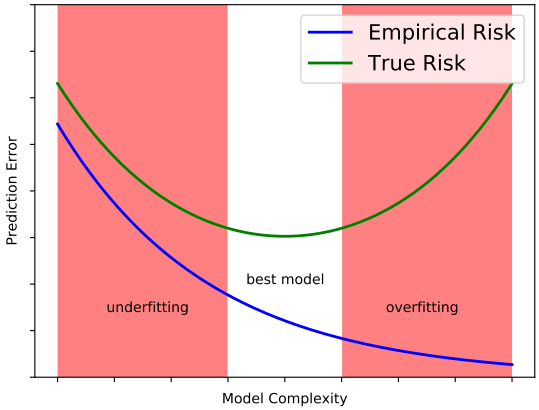

Figure 1.1: Representation of the bias-variance trade-off. On the left, the model complexity is low which makes the model too weak to learn well. This leads to underfitting. While on the right, the model complexity is high making the empirical risk low since the model is able to perfectly fit the training set. However, it can't generalize to new examples and thus has a high true risk. In the middle, a bias-variance trade-off is found yielding the best true risk reachable for the sample set available (w.r.t. the empirical risk).

We give an intuitive visual example in Figure 1.2. Three polynomial regression models aim to approximate the cosine function with some noise following a uniform distribution ($\epsilon \sim \mathcal{U}$), $\cos(\frac{3}{2}\pi x)+\epsilon = y$, by learning over a limited set of observations $(x_i, y_i)_{i=1}^{M}$. The only difference between these models is the degree of polynomial that they are allowed to have during training. We see that for a model with the highest complexity (the red curve), the training data points are perfectly predicted at the price of making huge errors in areas where there was no sample. This model is trying to learn the random noise ϵ. On the other hand, the green model underfits the data and is not able to capture the specificity of the target concept. By playing around with the model hyper-parameters (degree of the polynomial), we can find a model that tends to generalize well by finding the right trade-off in blue. However, "playing around" with the model complexity to find a model that generalize well can be a very difficult task.
In such a context, the Regularized Risk Minimization adds a regularization term and tries to find a trade-off between fitting the data and controlling the complexity of the model.

$$f = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{M} \ell(f(x_i)) + \lambda ||f||,$$

where $||f||$ is a norm and λ is a trade-off parameter which is basically determined empirically by cross-validation. This method is used to penalize complex methods

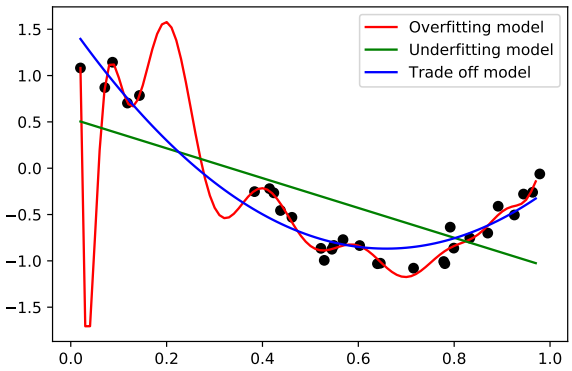

Figure 1.2: Example of regression with three different models.

to prevent overfitting. However, finding the right regularization is not trivial and depends on the task at hand.

Let us now we define more precisely the loss function, $\ell(\cdot, \cdot)$. Intuitively, it might be set to the 0/1 loss such as to assess the quality of a model using a classification error measure, as follows:

$$\ell_{0/1}(f(x), y) = \begin{cases} 1, \text{if} f(x) \neq y \\ 0, \text{otherwise.} \end{cases}$$

As simple as the 0/1 loss may seem, finding the minimizer of $\inf_{f \in \mathcal{F}} R(f)$ is difficult (NP-hard) mainly because of its non-differentiability, but also because of its non-convexity in f. Instead of using the 0/1 loss, we rather use convex surrogate loss functions. As mentioned earlier, there exist many different ones. We present the most common losses in Figure 1.3.

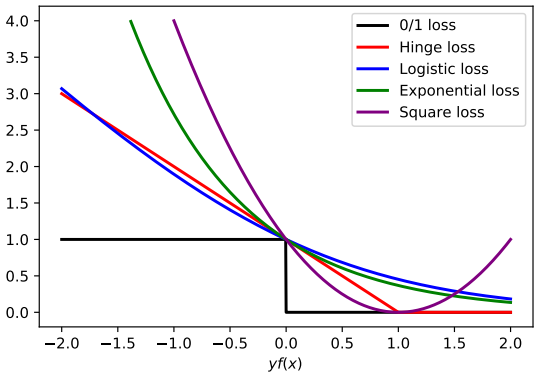

Figure 1.3: Main surrogate loss functions based on the *margin* $yf(x)$.

The hinge loss $\ell_{hinge}(f(x), y) = [1 - yf(x)]_+ = max(0, 1 - yf(x))$, mainly used in Support Vector Machines (SVM) (Cortes and Vapnik, 1995). This loss basically is 0 when y and $f(x)$ agree and is linearly increasing with $f(x)$ when they disagree. While it is not differentiable at 1, the hinge loss still has a subgradient with respect to the model parameters which is sufficient for the optimization.

The exponential loss $\ell_{exp}(f(x), y) = e^{-yf(x)}$, used for example in Adaboost (Freund and Schapire, 1997). This loss is a bit harder to optimize due to its exponential nature. Indeed, little variation in the model increases the loss exponentially. That being said, some learning methods are able to handle it quite effectively.

Logistic loss $\ell_{log}(f(x), y) = log(1 + e^{-yf(x)})$, which is ubiquitous in neural networks and also used in LogitBoost (Friedman et al., 2000). This loss function has a story where one wants to use $f(x)$ to estimate the probability of the associated label y, $P(y|x)$. In fact, this probability can be estimated with the logistic function (or the sigmoid),

$$P(y = +1|x) = \frac{1}{1 + e^{-f(x_i)}},\ P(y = -1|x) = \frac{1}{1 + e^{f(x_i)}}.$$

From this observation, one can compute the likelihood of the labels occurring in the training set,

$$\mathcal{L}\big((x_1, y_1), ..., (x_M, y_M)|f\big) = \prod_{i=1}^{M} \frac{1}{1 + e^{-y_i f(x_i)}} = e^{-\sum_{i=1}^{M} log(1+e^{-y_i f(x_i)})}.$$

Maximizing the likelihood is then equivalent to minimizing the logistic loss of this model. Apart from the fact that this loss is differentiable and convex which makes it a suitable loss for optimization, it has also bounded gradients (unlike the exponential loss) and leads to a better probability estimates.

Squared error $\ell_{se}(f(x), y) = (y - f(x))^2$, the well-known loss for the regression task. It is used for almost every single algorithm (Friedman, 2001) doing regression. In our context, we will use it in Gradient Boosting where our (weak) learner does not perform a classification but rather a regression task.

In this manuscript, we focus on ensemble learning that aims at combining different models $\{f_l\}_{l=1}^{L} \subseteq \mathcal{F}$ such that their combination outperforms the best single model $f_{best} \in \{f_l\}_{l=1}^{L}$:

$$R_{true}(\frac{1}{L}\sum_{l=1}^{L} f_l(x)) < R_{true}(f_{best}), \tag{1.2}$$

where L is the number of models being combined. The main point of ensemble learning is to take advantage of the diversity between the different models composing the ensemble. Of course, if they are too similar and do not complete each other in any way then building an ensemble with these models is not going to make our ensemble more robust. Thus, we need our models to bring some diversity and have a relevant empirical performance (*e.g.* better than random guessing). This last point differs between the ensemble method we use. In the following, we present different ensemble learning methods and their specificities.

1.2 Ensemble learning

We introduce here a key notion of this manuscript. Can we combine several models to increase the overall performance? In Section 1.1, we discussed how we could handle the bias-variance trade-off. It turns out that ensemble methods handle the bias-variance trade-off quite effectively while being relatively simple to train. We review three important methods that were the root of multiple algorithms.

At a time where decision trees (Quinlan, 1986; Breiman, 1984) were popular, they still suffered from handling the bias-variance trade-off very poorly. Indeed, a decision tree is a model that basically splits the dataset into two parts based on an information criterion (*e.g.* entropy) that defines how well the two resulting subsets (children nodes) perform compared to the parent. In practice, the number of nodes in a decision tree can go up to $\#nodes \leq \sum_{p=0}^{M} 2^p$ where p is the depth of the tree. Clearly, learning a tree where every single instance is correctly classified can lead to a very large tree that simply overfit the training set. Pruning methods (Quinlan, 1987) that aim at removing potential irrelevant nodes can be used but these techniques increase the complexity of the learning algorithm and are subject to arbitrary choices (Breiman, 1984). Ensemble learning appeared as a nice solution to build trees without too many human efforts (*e.g.* no manual pruning). Today, ensemble methods are used a lot to build efficient models and we will use them in Chapter 3 and 4. We now briefly describe these methods.

Bagging (Breiman, 1996) As in any ensemble method, bagging combines many classifiers by averaging their outputs into a final prediction. Its peculiarity is to train every classifier over a different subset $\mathcal{S}'$, drawn randomly from $\mathcal{S}$ such that $\mathcal{S}' \subset \mathcal{S}$. This technique is called *sampling with replacement* where the idea is to randomly draw examples from $\mathcal{S}$ and put them back in $\mathcal{S}$ such that the examples have a chance of being drawn multiple times. As we previously mentioned, trees are easy to train and have multiple advantages but their regularization was difficult

mainly because their growth without limit potentially leads to overfit the training dataset. Bagging offers a nice way to build trees without paying attention to how it overfits. Indeed, the randomness that we add using sampling makes trees more diverse and the final prediction much more robust to overfitting even if every single tree has a very high variance. The final bagging model is defined as follows:

$$F_{bagging}(x) = \frac{1}{T}\sum_{t=1}^{T} f_t(x).$$

Random forest (Breiman, 2001) (RF) is probably the most famous bagging method today. Inserting randomness during the training improved the results significantly. Indeed, we need models that bring a different, yet complementary knowledge to others. This is mainly done by (i) bagging the data (sampling with replacement) and (ii) selecting randomly subsets of features at each level of the tree.

Figure 1.4 presents how a RF makes it easy to handle the overfitting behaviour of a single tree and build a more general model.

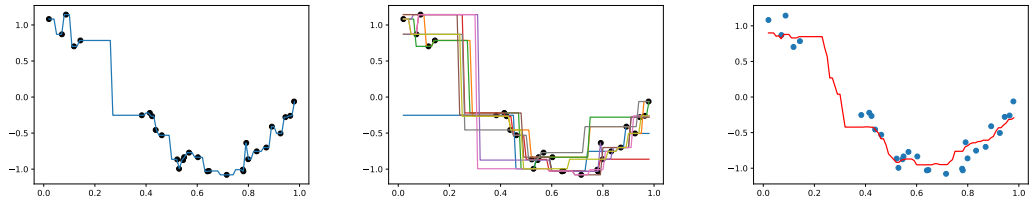

Figure 1.4: On the left, a single regression tree is completely overfitting the training set. In the middle, many different decision trees learned over different $\mathcal{S}' \subset \mathcal{S}$ are overfitting over the subsets. On the right, we plot the average of their outputs giving a single final decision.

Stacking (Wolpert, 1992) The idea of the stacking method comes from the observation that combining models linearly does not always yield the best solution. However, it is hard to combine different models efficiently since we don't know how they complement each other. A solution is to use a meta-learner that takes model outputs for every example and builds its own rules on how to combine them.

$$F_{stacking(x)} = \text{Meta}\big(f_0(x), f_1(x), f_2(x), ..., f_T(x)\big),$$

We will present a method inspired from stacking methods in Chapter 4.3.

Figure 1.5 gives an example of how many different learners can be used as the inputs of another model and yield better performance.

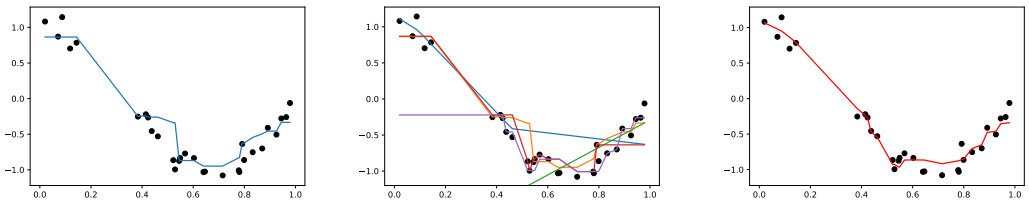

Figure 1.5: On the left, a single model is learned. In the middle, many different are learned with some diversity. On the right, a meta learner is taking their outputs as inputs and learns a better model.

Predicting on parts of the space where there is no training example is not an easy task. Ensemble methods like bagging and stacking offer a very simple and efficient way of finding models with a good generalization by combining many *different* models with a poor individual generalization. As we can see, the construction of the models composing the ensemble is made easy by willingly overfitting or underfitting. The combination of their outputs can also be very straight-forward as in bagging (simple linear combination). On the other hand, combining the model using stacking offers, potentially, an infinite number of combinations. Thus this technique is more prone to over-fitting than the others. A good compromise is the boosting approach that we present in the following section.

1.3 Boosting

The concept of boosting emerged with the work of Schapire (1990) which showed that, in theory, it is possible to improve the performance of any learner by combining a set of weak classifiers under the simple assumption that the base learner behaves better than random guessing. In machine learning, we often struggle to build a relevant model with high performances without falling into the overfitting scenario. And even without the overfitting problem, building a classifier to reach a high performance can be subject to computational constraints (i.e. infinite amount of data points). Boosting allows to alleviate the previous problems by only building *weak learners*. Two questions arise in this setting: 1) How do we use the information brought by each classifier? and 2) how can we build several weak learners such that they are complementary to solve the problem? In Freund and Schapire (1997), the authors develop the first and still so famous boosting algorithm, Adaboost (for **Ada**ptive **Boost**ing) which is a first approach to answer both questions.

The boosting process is basically focusing on the examples that were misclassified

by the already learned classifiers. In other words, boosting tends to learn general rules at the beginning and specializes to "difficult" examples along its T learning iterations. The final prediction in boosting is a weighted combination defined as follows:

$$F_{boosting}(x) = \sum_{l=1}^{T} \alpha_t h_t(x),$$

where, $\{\alpha_t\}_{t=1}^{T}$ represents the relative performance of the weak learners. For the rest of this manuscript, we write our T weak learners: $\{h_t\}_{t=1}^{T}$ to make our notations similar to that of the boosting community. We call them *weak hypotheses* or *weak learners*.

Figure 1.6 shows how boosting can create a strong model out of many weak and different learners.

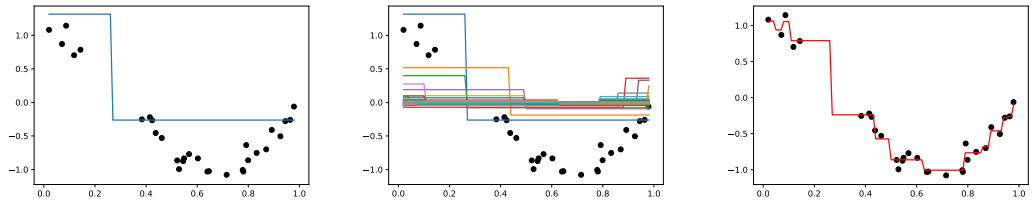

Figure 1.6: On the left, a single decision tree is underfitting. In the middle, many different decision trees learned using the boosting method (note that we plot here $\alpha_t h_t(x)$ instead of just $h_t(x)$). On the right, we plot their linear combination into a final decision.

Adaboost

Adaboost (Freund and Schapire, 1997) is the first boosting algorithm and initiated many algorithmic and theoretic research. This algorithm attracted many different fields due to its simplicity, performance and theoretical properties. To cite a few:

- Natural Language Processing (Abney et al., 1999; Carreras et al., 2003)
- Bioinformatics (Niu et al., 2006)
- Spam detection (Carreras and Marquez, 2001)
- Computer vision (Viola and Jones, 2001; Grabner and Bischof, 2006)
- Fraud detection (Viaene et al., 2004; Fan et al., 1999)

Adaboost operates as a rule of thumbs where, at each iteration, the most accurate rule to classify the dataset is found. Each new rule is built such as to focus more

on the examples that have been missed by the previous rules. By repeating the process for many iterations we finally end up with many rules that are combined linearly. The prediction is a weighted vote that defines the strong model.
In order to build diverse rules, the algorithm uses a weight $w_{i,t}$ for the example x_i at iteration t. Note that the number of iterations in boosting is defined by the number of models that compose the ensemble. The weight for a given observation varies along the training process and an asset of Adaboost is that this weight automatically adapts to the performance of the previous weak learners on this example. This is mainly why Adaboost is said to be adaptive. Moreover, every single rule h_t in the ensemble is given a weight α_t.

To accomplish all these steps, Adaboost is based on minimizing the exponential loss function ℓ_{exp}. The objective function can be written as follows:

$$R^{exp}(F_T) = \frac{1}{M}\sum_{i=1}^{M} \ell_{exp}(F_T(x_i), y_i) = \frac{1}{M}\sum_{i=1}^{M} e^{-y_i F_T(x_i)},$$

where $F_T(x) = \sum_{t=1}^{T} \alpha_t h_t$ and $h_t \in \{-1, 1\}$. All the losses presented in Section 1.1 could potentially be used, however, the choice of the exponential loss has several assets in this framework:

1. It opens the door to theoretical properties (Schapire and Singer, 1999).

2. It is very convenient to compute the weights for the dataset at each iteration.

3. It makes the optimal value α_t very easy to compute.

The main potential drawback of this loss function is the fact that it grows exponentially fast as F_t is wrong. In case of noise in the dataset, Adaboost could potentially spend a lot of effort classifying this noise correctly (Freund et al., 1999). In Dietterich (2000), the authors show that Adaboost is indeed susceptible to noise. Different solutions exist to cope with this problem such as the Brownboost (Freund, 2001) algorithm where the examples that are misclassified for too many iterations are left aside to let the learner focus on the remaining examples.
The specificity of Adaboost lies in the re-weighting schema used for all examples in the training set S with $\{w_{i,t}\}_{i=1}^{M}$, the weights, at each boosting iteration t. Note that we assume to have a base learner allowing to train with weighted samples. At iteration t, the weight $w_{i,t+1}$ is found such that it represents how well the strong learner $F_t(x_i)$ classifies x_i. This can be done using the following equation:

$$w_{i,t+1} = e^{-y_i F_t(x_i)}. \tag{1.3}$$

Mathematically, at iteration t we already computed $w_{i,t} = e^{-y_i F_{t-1}(x_i)}$. This allows us to have a simpler update for $w_{i,t+1}$:

$$w_{i,t+1} = e^{-y_i F_{t-1}(x_i)} e^{-y_i \alpha_t h_t} = w_{i,t-1} e^{-y_i \alpha_t h_t}.$$

Due to the exponential nature of the loss, the weights are normalized to get a statistical distribution such that $\sum_{i=1}^{M} w_{i,t+1} = 1$.

These weights are used to find a new h_t such that this weak learner minimizes the sum of the weights for the misclassified examples:

$$h_t = \operatorname*{argmin}_h \sum_{h(x_i) \neq y_i} w_{i,t}.$$

The second important point in Adaboost is to find the α_t that minimizes $R^{exp}(F_t)$ such that:

$$\alpha_t = \operatorname*{argmin}_\alpha \sum_{i=1}^{M} e^{-y_i (F_{t-1} + \alpha h_t)}.$$

The exponential loss and the assumption made on the output of the weak learners allow a very simple derivation to find a close form solution for which α_t is optimal.

$$\begin{aligned} R^{exp} &= \sum_{i=1}^{M} e^{-y_i F_{t-1}} e^{-y_i \alpha_t h_t} \\ &= \sum_{y_i = h_t(x_i)} e^{-y_i F_{t-1}(x_i)} e^{-\alpha_t} + \sum_{y_i \neq h_t(x_i)} e^{-y_i F_{t-1}} e^{\alpha_t}. \end{aligned}$$

It remains to find $\frac{\partial R^{exp}}{\partial \alpha_t} = 0$.

$$\begin{aligned} \frac{\partial R^{exp}}{\partial \alpha_t} &= \frac{\partial \left(\sum_{y_i = h_t(x_i)} e^{-y_i F_{t-1}(x_i)} e^{-\alpha_t} + \sum_{y_i \neq h_t(x_i)} e^{-y_i F_{t-1}} e^{\alpha_t} \right)}{\partial \alpha_t} = 0 \\ &= -e^{-\alpha_t} \sum_{y_i = h_t(x_i)} e^{-y_i F_{t-1}(x_i)} + e^{\alpha_t} \sum_{y_i \neq h_t(x_i)} e^{-y_i F_{t-1}} = 0 \\ \frac{e^{\alpha_t}}{e^{-\alpha_t}} &= \frac{\sum_{y_i = h_t(x_i)} e^{-y_i F_{t-1}}}{\sum_{y_i \neq h_t(x_i)} e^{-y_i F_{t-1}(x_i)}} \\ e^{2\alpha_t} &= \frac{\sum_{y_i = h_t(x_i)} e^{-y_i F_{t-1}}}{\sum_{y_i \neq h_t(x_i)} e^{-y_i F_{t-1}(x_i)}} \\ \alpha_t &= \frac{1}{2} \log \left(\frac{\sum_{y_i = h_t(x_i)} e^{-y_i F_{t-1}}}{\sum_{y_i \neq h_t(x_i)} e^{-y_i F_{t-1}(x_i)}} \right). \end{aligned}$$

Algorithm 1 Adaboost algorithm

1: Given: (x_i, y_i) where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$:
2: Initialize: $w_{i,1} = \frac{1}{M}$ for $i = 1, ..., M$.
3: **for** $t = 1$ to T **do**
4: Train $h_t : \mathcal{X} \to \{-1, +1\}$ using the weights w_t.
5: Choose $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$.
6: Update the weights:

$$w_{i,t+1} = \frac{w_{i,t} e^{-\alpha_t y_i h_t(x_i)}}{Z_t},$$

where Z_t is a normalization factor such that w_{t+1} is a distribution.
7: **end for**
8: Output the final model:

$$F^*(x) = sign(\sum_t^T \alpha_t h_t(x)).$$

Or as it is more commonly written in the literature:

$$\alpha_t = \frac{1}{2} \log(\frac{1 - \epsilon_t}{\epsilon_t}),$$

where $\epsilon_t = \sum_{h_t(x_i) \neq y_i} w_{i,t}$ is the weighted error rate of a given weak learner h_t. We summarize the steps of Adaboost in Alg. 1.

In Freund and Schapire (1997), the authors show that the empirical error (training error) is at most $R(F_{ada}) \leq e^{(-2 \sum_{t=1} \gamma_t^2)}$ where γ_t is the *edge* over random guessing of the t^{th} weak learner such that $\epsilon_t = \frac{1}{2} - \gamma_t$. This bound shows that the empirical error decreases exponentially fast along the boosting iterations. In the same paper, the authors also present how to bound the generalization error (true risk) of the final model in terms of its training error, the sample size M, the VC-dimension $\mathcal{V}$ and the number of boosting rounds T.

$$R_{true}(F_{ada}) < R(F_{ada}) + \mathcal{O}\left(\sqrt{\frac{T\mathcal{V}}{M}}\right)$$

This bound suggests that Adaboost will overfit as T becomes large. This has been debated in many papers as it was experimentally shown that Adaboost does not overfit even for thousands of rounds (Breiman, 1997; Drucker and Cortes, 1996). Moreover, it has been experimentally shown that, while the training error reaches 0, the test error still decreases which clearly contradicts the above bound. An explanation of this behaviour was made in Schapire et al. (1998) where the

authors give a different analysis using the *margin* of the training examples. The margin $m(x) \in [-1, +1]$ is defined to be

$$m(x) = \frac{y \sum_{t=1} \alpha_t h_t(x)}{\sum_{t=1} \alpha_t}.$$

The margin is positive if the model correctly classifies the example x and negative otherwise. This value can also be interpreted as the confidence of the model for a given example. With this value, the authors were able to derive a new bound defined as follows:

$$R_{true}(F_{ada}) < P(m(x) \leq \theta) + \mathcal{O}\left(\sqrt{\frac{\mathcal{V}_h}{M\theta^2}}\right)$$

for any $\theta > 0$ where $\mathcal{V}_h$ is the VC-dimension of the weak learner. This upper bound has the great advantage of not being dependent on the number of iteration T in its second term. It turns out that the first term was shown to be bounded above by

$$P(m(x) \leq \theta) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1 - \epsilon_t)}.$$

Adaboost is a very efficient algorithm but it only runs with the exponential loss which can be hard to handle in some cases (e.g. noisy datasets). In the following, we present a different boosting approach that uses a different loss function.

Additive Logistic Regression

This section presents the additive logistic regression algorithm (Logitboost) (Friedman et al., 2000) which uses the logistic loss instead of the exponential loss:

$$R^{log} = \frac{1}{M} \sum_{i=1}^{M} \ell_{log}(F_L(x_i), y_i) = \frac{1}{M} \sum_{i=1}^{M} \log(1 + e^{-y_i F_L(x_i)}).$$

Before explaining the Logitboost algorithm let us take a step back on its origins as it played a significant role in the evolution of boosting methods.

A simple linear model can be written in the following form:

$$F(x) = \alpha_0 + \alpha_1 x^1 + \alpha_2 x^2 + ... + \alpha_d x^d.$$

where $\{\alpha_t\}_{t=1}^{d}$ are the parameters of the model. It assumes that the problem can be explained as a linear combination of the input variables. However, in many cases and especially on real-life problems, a linear model is not expressive enough

to capture the full underlying concept of the data. To solve this problem, Additive Models (Friedman and Stuetzle, 1981) and its generalized version GAM (Hastie and Tibshirani, 1986) have been introduced. Instead of using a linear combination of the parameters, the goal is to use non-parametric functions that introduce non-linearity. The form of an additive models is as follows:

$$F(x) = f_1(x^1) + f_2(x^2) + ... + f_d(x^d).$$

This model is composed of different learners, one per feature. In order to update each function, the backfitting algorithm (Breiman and Friedman, 1985) is used to find f_j:

$$f_j = \underset{f}{\operatorname{argmin}} \frac{1}{M} \sum_{i=1}^{M} \left[y_i - \sum_{k \neq j} f_k(x_i^k) - f(x_i^j) \right]^2,$$

or in other words, f_j is updated such as to correct the error of the other functions. These functions only take a single dimension d of the whole feature input vector but more generally, we can assume that these functions use all inputs from x such that the model becomes more general:

$$F_t(x) = f_1(x) + f_2(x) + ... + f_t(x).$$

Finally, instead of having all the functions already defined, $F_t(x)$ can be updated in a greedy forward stepwise approach where a new model f_{t+1} is added:

$$f_{t+1} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{M} \left(y_i - F_t(x_i) - f(x_i) \right)^2.$$

At this step, we can see the connection with boosting where the $\{f_t\}_{t=1}^{T}$ would be the weak learners.

This backfitting algorithm was first made to work for regression tasks. In Hastie and Tibshirani (1986), the authors propose a new version of backfitting using the Newton-Raphson method for GAM and more specifically for the Additive Logistic Regression Model called Local Scoring and defined as:

$$F_t(x) = \frac{1}{1 + e^{-(f_1(x) + f_2(x) + ... + f_t(x))}} = P(y = 1|x) = p(x).$$

The Newton-Raphson method (Wallis et al., 1685; Raphson, 1697) is successively used as in the gradient descent algorithm to find the root of a function. The main difference is that it requires the second derivatives such that $\theta = \theta + \frac{\frac{\partial \ell(f(x), y)}{\partial \theta}}{\frac{\partial^2 \ell(f(x), y)}{\partial^2 \theta}}$ where θ is the parameter to update to find the root of $\ell(\cdot, \cdot)$. The higher order method

Algorithm 2 Logitboost algorithm

1: INPUT: A training set $\mathcal{S} = (x_i, y_i)_{i=1}^{M}$ where $x_i \in \mathcal{X}$ and $y_i \in \{0, 1\}$:
2: Initialize: $w_{i,1} = \frac{1}{M}$ and $p(x_i)$ for $i = 1, ..., M$, $F(x) = 0$.
3: **for** $t = 1$ to T **do**
4: Compute the newton step:

$$z_i = \frac{y_i - p(x_i)}{p(x_i)(1 - p(x_i))},$$

$$w_{i,t} = p(x_i)(1 - p(x_i)).$$

5: Fit the weak learner f_t by a weighted least-squares regression to z_i using weight $w_{i,t}$ for all $i \in 1, ..., M$.
6: Update $F(x) \leftarrow F(x) + \frac{1}{2} f_t(x)$ and $p(x) \leftarrow \frac{1}{1+e^{-F(x)}}$.
7: **end for**
8: Output the final model:

$$F^*(x) = sign(\sum_{t}^{T} f_t(x)).$$

allows to build better approximations and thus to converge in less iterations. However, it requires the second derivative which can be computationally expensive to have. Basically, the idea of Local Scoring is to find a new model f_{t+1} such that:

$$f_{t+1} = \underset{f}{\operatorname{argmin}} \left(F_t + \frac{y - p(x)}{p(x)(1 - p(x))} - F_t - f \right),$$

where $\frac{y-p(x)}{p(x)(1-p(x))}$ is the Newton-Raphson update. In fact, we only need f_{t+1} to approximate this update. The simplest strategy is to fit the new model over this update using a simple regression. Naturally, if the function perfectly fits the update, then no other step is required (all examples are well classified). However, in practice, we use weak learners as in Adaboost to handle the overfitting scenario. In Friedman et al. (2000), the authors actually use the same principle. The steps of LogitBoost are described in Alg. 2.

While the original LogitBoost algorithm is slightly different from Adaboost, in Collins et al. (2002) the authors a direct transformation to have an equivalence to Logitboost in the Adaboost framework with a single line modification in the Adaboost algorithm:

$$w_{i,k+1} = \frac{1}{1 + e^{y_i F_t}}. \tag{1.4}$$

Adaboost and Logitboost are both boosting algorithms that are built for specific losses. The former based on the exponential loss while the later works with the logistic loss. This last algorithm is in fact less constrained by its loss function: since the weights are based on the newton step, any other loss could potentially be used (as long as the second derivative is computationally feasible). In the following, we present the gradient boosting algorithm that basically generalizes the previous two methods to any loss function.

Gradient Boosting

The generic version of boosting for any loss function was first introduced by Breiman (1997) and later generalized by Friedman (2001). We previously presented the weights used in Adaboost and LogitBoost. In fact, Eq. 1.3 can be seen as the absolute gradient of $R^{exp}(F_t(x_i))$ in function of $F_t(x_i)$ such that

$$\left|\frac{\partial R^{exp}(F_t)}{\partial F_t(x_i)}\right| = \left|-y_i e^{-y_i F_t(x_i)}\right| = e^{-y_i F_t(x_i)} = w_{i,k+1},$$

and the same applies for Eq.1.4 in LogitBoost:

$$\left|\frac{\partial R^{log}(F_t)}{\partial F_k(x_i)}\right| = \left|\frac{-y_i}{1+e^{y_i F_k(x_i)}}\right| = \frac{1}{1+e^{y_i F_t(x_i)}} = w_{i,k+1}.$$

Indeed, the examples are weighted by the absolute value of the first derivative for the loss function used. At a given iteration t, we need to find h_t for a classification loss function ℓ_c as follows:

$$h_t = \underset{h}{\operatorname{argmin}}\, R^c(F_{t-1}+h) = \frac{1}{M}\sum_{i=1}^{M} \ell_c(F_{t-1}(x_i)+h(x_i), y_i). \tag{1.5}$$

Solving Eq.1.5 without assumption on the weak learner or the loss used is difficult. To solve this task, gradient boosting leverages regression algorithms to approximate the *negative gradients* also called the residuals such as to make a step toward the optimal solution of F_L in the function space (hence the name of functional gradient descent is often used for this method).

$$r_i^t = -\frac{\partial R^c(F_{t-1})}{\partial F_{t-1}(x_i)}.$$

As it was previously done in Logitboost, we want the weak learner h_t to minimize R^{sq}:

$$h_t = \underset{h}{\operatorname{argmin}}\, R^{sq}(F_{t-1}+h) = \underset{h}{\operatorname{argmin}}\, \frac{1}{M}\sum_{i=1}^{M}\Big(r_i^t - h(x_i)\Big)^2.$$

Finally, an advantage of boosting weak learners is the nice generalization behaviour of the final model. However, most of the time, we need to increase the complexity

Algorithm 3 Gradient boosting

1: INPUT: a training set $S = \{z_i = (x_i, y_i)\}_{i=1}^{M}$, a weak learner
2: Initialize $F_0(x) = 0$
3: **for** $t = 1$ to T **do**
4: Compute the residuals:

$$r_t^i = -\Big[\frac{\partial \ell(z_i, F_{t-1}(x))}{\partial F_{t-1}(x)}\Big], \forall z_i = (x_i, y_i) \in S \tag{1.6}$$

5: Fit a weak classifier (e.g. a regression tree) $h_t(x)$ to predict the targets r_t
6: Find $\alpha_t = argmin_\alpha \sum_{i=1}^{M} \ell_c(z_i, F_{t-1}(x_i) + \alpha h_t(x_i))$
7: Update $F_t(x)$ such that $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
8: **end for**
9: Output the final model:

$$F^*(x) = sign(\sum_{t}^{T} f_t(x)).$$

of these learners to reach better performance which also increases the risk of overfitting. One way to handle this is to regularize. In gradient boosting, we can mainly play with two parameters:

1. The learning rate λ which is a constant that shrinks the outputs of the weak learners $\sum_t^T \lambda \alpha_t h_t(x)$. This parameter is only used during training and can then be removed.

2. A parameter which basically imitates bagging and summarizes the stochastic gradient boosting developed in Friedman (2002). The idea is simply to take a subsample of the data for every weak learner. As in bagging, random successive subsampling helps to have a final model with a better generalization.

The steps of gradient boosting are summarized in Alg.3
Gradient boosting offers much more flexibility than other boosting algorithm based on Adaboost (i.e. any loss function can be used assuming we can find its first derivative and any base-learner doing regression is also a fit).

Note on the greedy approximation The greedy approximation of the residuals using a regression algorithm introduces an unwanted phenomenon. Indeed, when a weak learner is highly confident in its prediction and is correct regarding

the true label such that $h(x_i) >> 0$ for $y_i = 1$ or $h(x_i) << 0$ for $y_i = -1$ then, the classification loss tends to zero, $\ell_c(h(x_i), y_i) \to 0$. However, when the weak learner is being trained, it is highly penalized when its prediction is far from the residual r_i even if the sign of the prediction is correct and, therefore, actually minimizing the classification loss. This side effect of the greedy approximation can make the training of the weak learner more difficult since it is subject to more constraints than it should be. However, from a different point of view, penalizing too confident weak learners can also be in favour of the boosting algorithm which is above all a collaboration between different models. Indeed, as the prediction of $h(x_i)$ gets closer to r_i, the regression loss decreases. However, when this prediction goes too high and potentially makes the weak learner a decision-maker for the ensemble, the regression loss increases and penalizes the overconfidence.

The base learner For clarity, a base learner defines the algorithm boosted in the model while the weak learners define the models built during the training process. So far, we have not made any assumption on the base learner simply because any learner is acceptable in the theoretical framework of boosting. Even a strong learner could be used at the risk making the model prone to overfitting. Note that, in practice, respecting the weak assumption ($\gamma_t > \frac{P}{M}$, where $\frac{P}{M}$ is the accuracy of a random classifier) is not mandatory since the weak learner weight, if correctly computed, should appropriately switch the signs of the predictions if the weak learner is less than the random guess and discredit the prediction of this weak learner by lowering the weight to 0. That being said, since its invention, trees have proved to be much more efficient than other base-learners (Schapire and Singer, 1999; Friedman, 2001, 2002; Freund et al., 1996) (e.g. naive bayes, perceptron, ...). Moreover, trees are very straight forward learning algorithms with different assets:

1. They can be learned with sample weights rather easily (i.e. by using these weights in the splitting criterion)
2. They are invariant on the input variables (i.e. no pre-processing required to scale the continuous features).
3. They are easily distributed which improves a lot the computation time.
4. The internal rules are human-readable which is a rare commodity in machine learning algorithms to address the problem of *interpretable AI*.
5. The Random Forest bagging method (Breiman, 2001) proved to be a very good algorithm on real datasets and also uses trees.

We formally define a tree structure as:

$$h_{\text{tree}}(x; \{b_j, L_j\}_1^J) = \sum_{j=1}^{J} b_j \mathrm{I}(x \in L_j),$$

where $\{L_j\}_{j=1}^{L}$ are different final leaves of the tree that cover the entire space of x and $\mathrm{I}(\cdot)$ is the indicator function that takes the value one if $\cdot$ is true and 0 otherwise. In gradient boosting, the trees used are doing a regression over the residuals where b_j is the value of the region R_j. Note that, in case of stumps, we only have $J = 2$ (decision tree with a single split) thus only two values define the entire input space. The training process of a decision tree is a greedy learning algorithm that tests every possible split in the dataset over the $x^i \forall x \in S, i \in \{1, 2, ..., d\}$ and, in general, every parent node only splits into two parts. In order to find these splits, we have a criterion that allows us to evaluate the quality of a split for a given node containing $\{x_i, y_i\} \in S_{\text{node}} \subset S$. In gradient boosting, regression trees are used and the splitting criterion is defined as follows:

$$\mathbf{v}_{\text{split}} = \underset{x_a^d}{\text{argmax}} \frac{|S_{\text{L}}|}{|S_{\text{node}}|} \sum_{x_i^d < x_a^d} (r_i - \bar{r}_L)^2 + \frac{|S_{\text{R}}|}{|S_{\text{node}}|} \sum_{x_i^d \geq x_a^d} (r_i - \bar{r}_R)^2 - \sum_{x_i} (r_i - \bar{r})^2 , \tag{1.7}$$

where $|S_{\text{L}}| = \sum_{x_i^d < x_a^d} 1$, $|S_{\text{R}}| = \sum_{x_i^d \geq x_a^d} 1$, $\bar{r}_L = \frac{\sum_{x_i^d < x_a^d} r_i}{|S_{\text{L}}|}$, $\bar{r}_R = \frac{\sum_{x_i^d \geq x_a^d} r_i}{|S_{\text{R}}|}$ and $\bar{r} = \frac{\sum_{x_i} r_i}{\sum_{x_i} 1}$. $\mathbf{v}_{\text{split}}$ is the feature value that best splits the entire dataset at the node. Note that in Eq. 1.7, $\sum_{x_i} (r_i - \bar{r})^2$ is only used for the stopping criterion. Indeed, the regression loss cannot be improved in the children's nodes then there exists no x_a^d such that

$$\frac{|S_{\text{L}}|}{|S_{\text{node}}|} \sum_{x_i^d < x_a^d} (r_i - \bar{r}_L)^2 + \frac{|S_{\text{R}}|}{|S_{\text{node}}|} \sum_{x_i^d \geq x_a^d} (r_i - \bar{r}_R)^2 - \sum_{x_i} (r_i - \bar{r})^2 > 0,$$

and the boosting algorithm can be stopped. In practice, we might allow this behavior and thus remove the stopping criterion.

Earlier, we mentioned some constraints induced by the regression algorithm fit over the residuals. In fact we can slightly modify Eq. 1.7 such as to remove the constraints by simply maximizing the sign of the residuals in the two children nodes as follows:

$$\mathbf{v}_{\text{split}} = \underset{x_a^d}{\text{argmax}} \frac{|S_{\text{L}}|}{|S_{\text{node}}|} \left(\sum_{x_i^d < x_a^d} r_i \right)^2 + \frac{|S_{\text{R}}|}{|S_{\text{node}}|} \left(\sum_{x_i^d \geq x_a^d} r_i \right)^2 - \left(\sum_{x_i} r_i \right)^2 . \tag{1.8}$$

Eq. 1.8 is induced from the work of Mason et al. (2000) in which they define a new algorithm called Anyboost where they propose to find the best h_t as follows:

$$h_t = \underset{h}{\operatorname{argmax}} \frac{1}{M} \Big(\sum_{i=1}^{M} r_i^t \cdot h(x_i) \Big)^2.$$

In the following section, we present a recent gradient boosting algorithm with a similar approach to compute the best splitting value. Note that this modification might not be easily feasible for any weak learner.

Another advantage of using trees as weak learners in gradient boosting is the fact that the weight α can be computed at the leaf level instead of the entire tree. This implies that we have multiple weights for one weak learner equal to the number of terminal leaves $\{\alpha_t^j\}_j^J$. This has a strong advantage over using a single weight because different terminal leaves do not yield the same predictive performance and therefore should be weighted differently. In fact, other weak learners such as perceptron, naive bayes or neural network could benefit from such weighting schema. In Chapter 4.3, we generalize this weighting schema to different weak learners.

In the following section, we detail a variant of gradient boosting which improved the training process in terms of computation speed but also generalizes the Newton-Raphson method used in Logitboost to different loss functions assuming trees as weak learners.

Extreme Gradient Boosting

Extreme Gradient Boosting (Chen and Guestrin, 2016), also called XGBoost, is a competitive machine learning algorithm by its efficiency and its flexibility. There are some distinct differences with the original gradient boosting algorithm that basically aim to get a better generalization. First, the set of hyper-parameters in XGboost is much bigger than in the classical gradient boosting algorithm. An important point in XGBoost is that, instead of using the objective function directly, it uses a second order Taylor approximation. We present this objective function in Eq. 1.9. The constraint is only to be able to compute the first and second order derivatives for a given loss function.

$$\begin{aligned} R &= \sum_{i=1}^{M} [\ell(F_{t-1}(x_i) + h_t(x_i), y_i)] + \omega(h_t) \\ &\approx \sum_{i=1}^{M} [\ell(F_{t-1}, y_i) + \mathbf{g}_i h_t(x_i) + \frac{1}{2} \mathbf{h}_i h_t^2(x_i)] + \omega(h_t) = R^{taylor}. \end{aligned}$$

Note that now $\ell(F_{t-1}, y_i)$ is a constant in the objective function and can be removed. We end up with the following simplified objective function:

$$R^{\text{taylor}} = \sum_{i=1}^{M} [\mathbf{g}_i h_t(x_i) + \frac{1}{2}\mathbf{h}_i h_t^2(x_i)] + \omega(h_t), \tag{1.9}$$

where $\mathbf{g}_i = \frac{\partial \ell_c(F_{t-1}(x_i), y_i)}{\partial F_{t-1}(x_i)}$, $\mathbf{h}_i = \frac{\partial^2 \ell_c(F_{t-1}(x_i), y_i)}{\partial^2 F_{t-1}(x_i)}$ and $\omega = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{J} b_j^2$ with γ and λ two regularization terms. This Taylor approximation of the objective function offers a nice property that was introduced in Friedman et al. (2000) where they use a Newton update as the optimal value for a given leaf. Indeed, we can rewrite Eq. 1.9 with the new tree h_{tree} such as to minimize the loss function.

$$\begin{aligned}
R^{\text{taylor tree}} &= \sum_{i=1}^{M} [\mathbf{g}_i h_{\text{tree}}(x_i) + \frac{1}{2}\mathbf{h}_i h_{\text{tree}}^2(x_i)] + \omega(h_{\text{tree}}) \\
&= \sum_{j=1}^{J} \left[\sum_{x_i \in L_j} [\mathbf{g}_i b_j + \frac{1}{2}\mathbf{h}_i b_j^2] \right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{J} b_j^2 \\
&= \sum_{j=1}^{J} \left[(\sum_{x_i \in L_j} \mathbf{g}_i) b_j + \frac{1}{2}(\sum_{x_i \in L_j} \mathbf{h}_i + \lambda) b_j^2 \right] + \gamma T.
\end{aligned}$$

It only remains to find the optimal value b_j^* for a leaf by solving $\frac{\partial R^{\text{taylor tree}}}{\partial b_j^*} = 0$.

$$\begin{aligned}
\frac{\partial R^{\text{taylor tree}}}{\partial b_j^*} &= (\sum_{x_i \in L_j} \mathbf{g}_i) + (\sum_{x_i \in L_j} \mathbf{h}_i + \lambda) b_j = 0 \\
b_j^* &= -\frac{\sum_{x_i \in L_j} \mathbf{g}_i}{\sum_{x_i \in L_j} \mathbf{h}_i + \lambda} = \frac{\sum_{x_i \in L_j} r_i}{\sum_{x_i \in L_j} \mathbf{h}_i + \lambda},
\end{aligned}$$

where b_j^* is a Newton Raphson step. Thus we can compute the corresponding objective function for h_{tree} by plugging this optimal leaf value in $R^{\text{taylor tree}}$and taking $r_i = -\mathbf{g}_i$:

$$R^{\text{taylor tree}} = \frac{1}{2} \sum_{j=1}^{J} \left[\frac{(\sum_{x_i \in L_j} r_i)^2}{\sum_{x_i \in L_j} \mathbf{h}_i + \lambda} \right] + \gamma T. \tag{1.10}$$

Eq.1.10 gives us the total error of the tree considering all terminal leaves. At a given node, the best splitting value $\mathbf{v}_{\text{split}}$ can be found as follows:

$$\mathbf{v}_{\text{split}} = \underset{x_a^d}{\text{argmax}} -\frac{1}{2} \left[\frac{\left(\sum_{x_i^d < x_a^d} r_i\right)^2}{\sum_{x_i^d < x_a^d} \mathbf{h}_i + \lambda} + \frac{\left(\sum_{x_i^d \geq x_a^d} r_i\right)^2}{\sum_{x_i^d \geq x_a^d} \mathbf{h}_i + \lambda} - \frac{\left(\sum_{x_i} r_i\right)^2}{\sum_{x_i} \mathbf{h}_i + \lambda} \right] - \gamma. \tag{1.11}$$

In summary, as we previously mentioned, boosting algorithms aim at finding a new h_{new} such that:

$$h_{\text{new}} = \underset{h}{\text{argmin}} \sum_{i=1}^{M} \ell_c(F_t + h, y). \tag{1.12}$$

For Adaboost, and gradient boosting, it boils down to finding a h_{new} such that it approximates the residuals. For Logitboost and XGboost, the new weak learner approximates the Newton-Raphson update such that:

$$h_{new} \approx -\frac{\frac{\partial \ell(F_t,y)}{\partial F_t}}{\frac{\partial^2 \ell(F_t,y)}{\partial^2 F_t}}.$$

The classical gradient boosting and XGboost differ mainly in the splitting criterion when building the weak learner (e.g. Eq.1.7 for gradient boosting and Eq.1.11 for XGboost). That being said, it is hard to compare them since they both have advantages and drawbacks. First, gradient boosting approximates the residuals using the mean squared loss while XGboost finds an optimal solution to the taylor approximation of objective function. XGboost approximation allows to quickly find the optimal splits and values of the trees that seem more intuitive even if the optimal values found are based on a Taylor approximation and thus are not optimal regarding the true objective function. Despite the recent fame of XGboost framework, in this work, we found some constraints to use XGboost. While gradient boosting only needs the objective function to be differentiable, XGboost needs it to be twice differentiable and different from 0. This leads to another "issue" that is, XGboost only works for strictly convex objective functions while it is not required for gradient boosting. As we will see in Chapter 3, we sometimes need to get rid of the convexity constraint to reach a more specific goal (e.g. optimizing the top rank).

		Actual class: Positive	Actual class: Negative	Total
Predicted class	Positive	TP	FP	$\sum_{i=1}^{M} \mathrm{I}(f(x_i) = 1)$
	Negative	FN	TN	$\sum_{i=1}^{M} \mathrm{I}(f(x_i) = -1)$
	Total	P	N	M

Table 1.1: Confusion matrix with the number of True Positives (TP), True Negative (TN), False Positive (FP) and the False Negative (FN).

1.4 Class Imbalance Learning

In this section, we describe the class-imbalance problem that has been repeatedly reported in the literature (Chawla et al., 2004; He and Ma, 2013; He and Garcia, 2008; Kubat et al., 1997) and present different solutions from the state of the art. As stated earlier, we focus on the binary supervised learning setting with $y \in \{-1, 1\}$. In imbalanced scenarios, $y = 1$ often describes the minority (positive) class while $y = -1$ represents the majority (negative) class. Let P (resp. N) be the number of positive (resp. negative) examples such that $P + N = M$. In this setting, we rewrite the training set $\mathcal{S}$ such that $\mathcal{S}^+ = \{z_i^+ = (x_i^+, y_i^+) | y_i = +1\}_{i=1}^{P}$ and $\mathcal{S}^- = \{z_i^- = (x_i^-, y_i^-) | y_i = -1\}_{i=1}^{N}$ where $\mathcal{S}^+ \cup \mathcal{S}^- = \mathcal{S}$. We define the imbalance ratio as $\rho = \frac{N}{P}$ and the proportion of examples in the minority class as $\pi = \frac{1}{1+\rho} = \frac{P}{M}$

1.4.1 Evaluation metrics

The evaluation metric is a rather important part of machine learning since, depending on the selected criterion, different models are preferable. We first describe the well-known *confusion matrix* in Table 1.1, that contains four standard measures in classification:

1. True Positives (TP), the number of positive examples correctly classified.
2. False Positives (FP), the number of misclassified negative examples.
3. True Negatives (TN), the number of negative examples correctly classified.
4. False Negatives (FN), the number of misclassified positive examples.

$$\left.\begin{array}{l} \mathrm{TP} = \sum_{i=1}^{P} \mathrm{I}(f(x_i) = 1) \\ \mathrm{FN} = \sum_{i=1}^{P} \mathrm{I}(f(x_i) = -1) \end{array}\right\} \text{Positive examples}$$

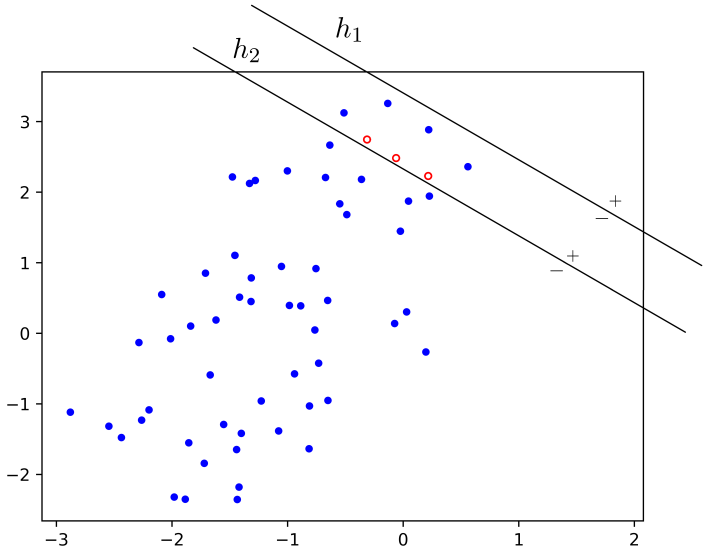

Figure 1.7: Comparison of different classifiers on an imbalanced toy dataset in two dimensions.

$$\left.\begin{array}{l}\text{TN} = \sum_{i=1}^{N} \text{I}(f(x_i) = -1) \\ \text{FP} = \sum_{i=1}^{N} \text{I}(f(x_i) = 1)\end{array}\right\} \text{Negative examples.}$$

We mentioned previously that the 0/1 loss was directly related to the accuracy which can be written with the previous terms:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP+TN+FP+FN}}, \tag{1.13}$$

or in other words, the percentage of correctly classified instances. It is clear that in the case where $P >> N$ or $N >> P$, the minority class is under-represented. Figure 1.7 illustrates this problem using a toy dataset over which we learn two linear classifiers, h_1 and h_2 over two classes: the blue class and the red class (in minority). In this example, h_1 only makes 3 errors compared to h_2 that misclassifies 4 examples. From the accuracy point of view, h_1 is better. However, h_1 classifies every example as negative which makes it a poor classifier not able to predict any positive example where h_2 correctly classifies all the minority class (red) at the price of less accuracy (more false positive, FP). From this example, we see that the accuracy can be irrelevant in the class imbalanced setting. In the literature, this problem has been observed many times (Guo and Viktor, 2004; Weiss, 2004; Chawla et al., 2003; Sun et al., 2007). We need different metrics to assess the quality of the models in terms of classification that we present in the following.

Precision defines the percentage of well-classified positive examples (minority class) over the total number of examples classified in the same class.

$$\text{precision} = \frac{\text{TP}}{\text{TP+FP}}, \tag{1.14}$$

with $precision \in [0, 1]$ and where $precision = 1$ is when there is no false positive example.

Recall defines the percentage of retrieved positive examples (from the minority class).

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{1.15}$$

with $recall \in [0, 1]$ where $recall = 1$ is the best value where all examples from the positive class are well-classified.

Alone, Precision (Eq. 1.14) or Recall (Eq. 1.15) do not inform enough to make any conclusion on the classifier performance simply because we can have a precision close to 1 with a recall close to 0 and vice versa. We rather use metrics that combine both of them.

F_β **score** is the weighted harmonic mean between precision and recall. As to offer more flexibility, we can use β to emphasize more on precision or recall.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2) \cdot \text{precision} + \text{recall}}, \tag{1.16}$$

with $F_\beta \in [0, 1]$ where $F_\beta = 1$ is the best achievable value (perfect classifier).

Matthews Correlation Coefficient (MCC) is the geometric mean

$$MCC = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}, \tag{1.17}$$

with $MCC \in [-1, 1]$ where $MCC = 1$ is the best achievable value (perfect classifier).

These metrics are subject to debates in the literature to know which one is the best. In Chicco (2017), the authors claim that the F_β score could be overoptimistic in case where the number of positive examples is much higher than the number of positive examples. Indeed, since the F_β focuses on the class of interest (the positive class), having an imbalanced dataset with more positives than negatives would highly affect this metric by increasing it while the model could be very bad on the negative class. The authors then claim that the MCC score, not focused

on a particular class is a better metric for imbalance problems. However, nothing prevents the F_β score from being focused on the negative class if the previous case arises. There also exist different ways of computing precision, recall, the F_β score and MCC that are mainly used in multi-class settings:

- **Standard**: only compute the metric over the class of interest.
- **Average**: compute the metric for each class and take their mean.
- **Micro**: compute the TP, FP, FN, and TN for each class and sum them up respectively to obtain a final number for each measure (unsuitable in the class imbalance setting)
- **Weighted average**: similar to the Average but uses a weight for each class.

While the Average, Micro and Weighted average methods can be good in multi-label classification, their usefulness in binary class datasets is rather limited. Indeed, in this setting, we can assume that the classifier performance on the majority class is always better (since it is biased toward it) and thus we can focus only on the metrics for the minority class which is the class of interest. Moreover, in the binary setting, the confusion matrix offers enough information on the majority class whereas in the multi-class setting all the classes other than the focused one are mixed together. For these reasons, in the following, we only use the Standard method.

Measuring the potential of a model

We now look at the model performance evaluation from another angle. So far we assumed that f is a function $f : \mathcal{X} \to \mathcal{Y}$ but most learning algorithms naturally output a score before actually predicting a class. It feels then more natural for these algorithms to write $f : \mathcal{X} \to \mathcal{Z}$ where $\mathcal{Z}$ is a decision space. In the rest of this manuscript, we assume that $z \in \mathcal{Z} \subseteq \mathbb{R}$, $P(y = 1|z) \xrightarrow{z \to +\infty} 1$ and $P(y = -1|z) \xrightarrow{z \to -\infty} 0$. Now that we defined the new decision space, we write a function f^* such that

$$f^*(x) = \begin{cases} +1, \text{if} f(x) > \tau \\ -1, \text{otherwise.} \end{cases}$$

where τ is the *decision threshold*. This new parameter τ offers the possibility to potentially create an infinite number of classifiers since for every value of τ the classifier predictions change. Based on this observation, we can review how we assess a model given its output prediction scores.

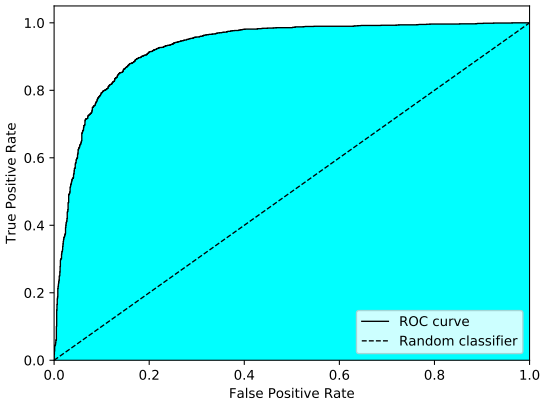

Figure 1.8: Example of ROC curve.

The main drawback of the previous evaluation metrics is that they assess only one level of decision (one decision threshold). For a given classification problem there is no reason for this decision threshold to be the best one. This is all the more true when the data are imbalanced. Figure 1.7 illustrates our previous statement. h_1 is going to be the resulting classifier when using a classical linear classifier, however, by modifying the decision threshold the resulting model is much more relevant (h_1 can potentially be equal to h_2). In fact, the decision threshold is often biased toward the majority class which often leads to undesirable classification.

A popular metric to assess the model performance over all possible decision thresholds and thus estimate the "potential" of a model is the Area Under the Receiver Operator Characteristic curve (AUCROC). Instead of using the quantity directly from the confusion matrix, the ROC curve uses the True Positive Rate (TPR= $P(f(x^+) > \tau)$ =Recall (Eq. 1.15)) and the False Positive Rate (FPR = $\frac{\text{FP}}{\text{FP+TN}} = P(f(x^-) > \tau)$). The use of these probabilities instead of the quantities makes the ROC curve insensitive to the class imbalance.

$$\begin{aligned} \text{AUCROC} &= \int_0^1 P(f(x^+) > \tau) dP(f(x^-) > \tau) \\ &= \int_{-\infty}^{+\infty} P(f(x^+) > \tau) \frac{\partial P(f(x^-) > \tau)}{\partial \tau} d\tau \\ &= P(f(x^+) > f(x^-)). \end{aligned} \tag{1.18}$$

We give an example of the ROC curve in Figure 1.8.

The last evaluation metric that we present is closely related to the precision (Eq. 1.14) and recall (Eq. 1.15). However, instead of computing these metrics for one predefined decision threshold, we compute them for all relevant decision

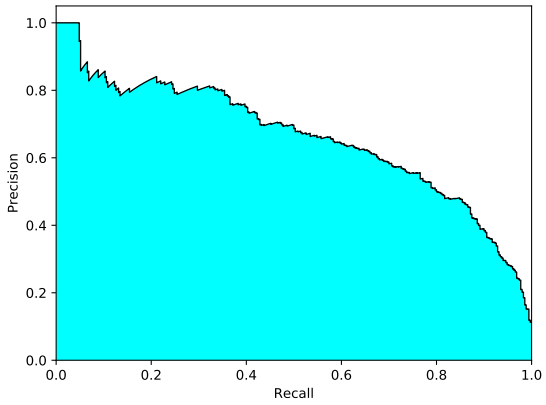

Figure 1.9: Example of the precision and recall curve.

threshold and average them. This metric is called the Area Under the Precision and Recall Curve (AUCPR) or the average precision.

$$AUCPR = \int_0^1 P(y = 1|f(x^+) > \tau)dP(f(x^+) > \tau) \tag{1.19}$$

We give an example of the precision and recall curve in Figure 1.9.
In Chapter 3, we present AUCROC and AUCPR in more details and provide a smooth objective function derived from AUCPR.

1.4.2 Sampling Methods

When dealing with class imbalance learning, a standard solution consists in resorting to sampling methods. The idea is to rebalance the dataset such that both classes are well-represented in the training dataset. These methods are mainly based on either removing examples from the majority class (*undersampling*) and/or increasing the examples of the minority class (*oversampling*). For the following, we remind that $\mathcal{S}^+$ defines our minority class and $\mathcal{S}^-$ the majority class.

Random Undersampling Used in the early ages of machine learning (Breiman, 1984), the idea is to randomly remove examples from $\mathcal{S}^-$ such that the minority class gets more importance in the eyes of the learning algorithm. We want $\mathcal{S}^{-*} \subset \mathcal{S}^-$ where $\mathcal{S}^{-*}$ is our new set of examples from the majority class. This method is rather straightforward and has the advantage of reducing the training time by decreasing the number of examples over which the algorithm is learning. However, a strong negative point of this method is the potentially huge amount of information that might be lost by removing those examples which makes the right imbalance ratio hard to find.

Algorithm 4 SMOTE algorithm (Chawla et al., 2002)

INPUT: S^{min}, k the number of neighbours to consider
for $i = 1$ to $|S^{min}|$ **do**
  Compute the k-NN set $\{x_n\}_{n=1}^{k}$ for x_i and choose a random neighbour $\hat{x}_i \in \{x_n\}_{n=1}^{k}$
  Compute the distance vector $dist(\hat{x}_i, x_i)$
  Multiply distance vector by a random number $\delta \in [0, 1]$
  $x_{new} = x_i + \delta\, dist(\hat{x}_i, x_i)$
  Add x_{new} to S^{min}
end for

Tomek links (Tomek, 1976) As random undersampling suffers from removing relevant information, Tomek links removes examples from the majority class by selecting those that are the closest to the minority class. In other words, we remove the examples from the majority class, starting by $x_{remove} = \text{argmin}_{x_i}\, dist(x_i, x_j)$ where $i \in \{1, ..., |\mathcal{S}^-|\}$, $j \in \{1, ..., |\mathcal{S}^+|$ and $dist$ is a distance (*e.g.* Euclidean distance, $dist = (x_i - x_j)^2$). This x_{remove} is called a Tomek link. The process is repeated until user satisfaction. Note that this method is very computationally expensive as $M \to +\infty$.

Random Oversampling This is a really naive way to increase the number of examples in the minority class $\mathcal{S}^+$ such that $\{x_i, y_i\}_{i=1}^{O} \sim \{x_i, y_i\}_{i=1}^{P} \in \mathcal{S}^+$ where $O > P$. However, such oversampling method is prone to overfitting in case of extreme class imbalance ratio.

SMOTE Synthetic Minority Oversampling Technique is a method to create new examples Chawla et al. (2002) from the minority class in a clever way such that there is also a certain diversity between the examples. This method is introduced in Chawla et al. (2002) where, based on a k-NN algorithm (Altman, 1992), they create new synthetic and diverse examples. This advanced oversampling technique is presented in Alg. 4.

As well as for undersampling, many different methods were then introduced based on SMOTE such as Borderline-SMOTE (Han et al., 2005) or ADASYN (He et al., 2008) and hybrid methods that we present later. Note that, in the literature, SMOTE is used with only one random number for the whole feature vector. However, in Chawla et al. (2002), the algorithm presenting SMOTE has a random number for each feature. These two versions differ mainly in the space where they add new synthetic examples. The version using only one random number adds new examples on the line connecting the two examples from the minority class.

		Actual class	
		Positive	Negative
Predicted class	Positive	$C(1,1)$	$C(-1,1) = c_{-1}$
	Negative	$C(1,-1) = c_1$	$C(-1,-1)$

Figure 1.10: Cost matrix

For the version where a random number is created per feature, the new examples are added in the hypercube where the two examples are the opposite vertices. There is no study on these two methods and the original paper does not clearly state which one it uses. However, we believe that using a single line to create new examples is not enough for high dimensions of the vector x.

When the dataset contains categorical features the distance between two points is not straightforward. Indeed, Alg. 4 only works for continuous features. In a small variant of SMOTE, the authors propose to penalize the distance between two points based on how many times their categorical features differ. In practice the authors propose to add the median of the standard deviation of all continuous features. In other words, the distance between two points is increased by a certain value multiplied by the number of times their categorical features differ.

1.4.3 Cost-Sensitive Learning

Sampling methods are a way to balance the dataset when it is very skewed. However, they have main drawbacks: they remove potentially relevant information or they add new examples that should not exist. An alternative to a sampling method is to weigh the examples during the training such that the minority class gets more importance. To present this kind of learning, we can redefine our cost matrix as presented in Table 1.4.3 (similarly to Table 1.1). Note that we don't usually set a cost on well classified instances. The application of these costs can be straight forward using the objective function R:

$$R^{cost} = \sum_{i=1}^{P} \ell_c(f(x_i), 1)c_1 + \sum_{i=1}^{N} \ell_c(f(x_i), 0)c_{-1}. \tag{1.20}$$

Eq 1.20 allows the user to give more importance to the minority class. Now, in some specific applications, examples from the minority class are not equally important. For that reason, one can redefine the costs such that each example is weighted by a relative importance. For example, in fraud detection, one could assume that examples with the highest amount are more important than the others. As we will present in Chapter 2, while it seems that this method is suited for fraud

detection, there are many constraints from the real life that make the cost very hard to find.

1.4.4 Threshold learning

We presented before that most learners are based on a decision threshold where $f^*(x_i) = 1$ if $f(x_i) > \tau$ and $f^*(x_i) = -1$ if $f(x_i) \leq \tau$. In fact, the decision threshold τ is implicit in most learning algorithms. For example, a perfect boosting model gives $F(x^+) > 0$ for positive examples where $F(x^-) < 0$ for negative examples. The decision threshold is simply set at 0. However, in case of imbalance learning, this implicit value τ can be disastrous. For example, a basic classifier learned over a dataset where $\rho = 1000$ is highly biased toward the negative class with a high risk of having $f(x_n) = 0$, $\forall x_n \sim \mathcal{D}$ which yields an accuracy of 99.9% but is useless from the positive class perspective. In fact, even changing the distribution using sampling or cost-sensitive learning techniques may give a misleading implicit decision threshold (Dal Pozzolo et al., 2015c; Provost; Yu et al., 2016). This section presents threshold learning methods which assume that the classifier already has a good knowledge of the task but the best decision threshold is still to be found. We give an intuitive example in Figure 1.7.

In the previous evaluation metrics, the decision threshold τ is not fixed such as ROC (Eq. 1.8) and AUCPR (Eq. 1.9). In fact, these curves can help us to decide what decision threshold is the best. A straightforward approach is to have a holdout set over which we compute the previous curves and pick the best threshold. The reason why this is done experimentally is that every real life classification task, and not only the ones that suffer from the class imbalance problem, are under strong user preferences. Clearly, depending on the problem, one would prefer precision over recall when another would do the opposite. We give more precise examples in Chapter 2.

The goal of the thresholding method is to optimize a decision threshold dependent evaluation metric such as the F_1 score. Many different research highlighted the need to use this technique, especially when sampling or cost-sensitive methods were used (Dal Pozzolo et al., 2015c,b). In Parambath et al. (2014) the authors present an optimization method based on cost-sensitive learning to maximize the F_1 score and even then, the authors advise adjusting, a posteriori, the threshold based on the classifier scores. Their claim is that by optimizing the F_1 score, the knowledge of the classifier is better suited.

1.4.5 Ensemble learning for the class imbalance setting

So far we presented general methods that could be applied to most learning algorithms. We already presented ensemble methods whose idea is to combine a set of classifiers to achieve better performance. In fact, there exist different ensemble methods that are dedicated to address the imbalanced case scenario.
Balance Cascade and Easy Ensemble (Liu et al., 2009) are two ensemble methods created specifically to answer this problem. As we stated earlier, while undersampling has a strong advantage in terms of computation by removing parts of the dataset, it also discards potential relevant information and alters the real class distribution. These two ensemble methods use sampling but also try to answer the previous problem.

Easy Ensemble In (Liu et al., 2009), the authors present this method which is very close to bagging. Instead of sampling randomly $\mathcal{S}' \subset \mathcal{S}$, a new classifier h_k is learned over a balanced set where $\mathcal{S}^+ \cup \mathcal{S}_k^- \subset \mathcal{S}^-$ such that $|\mathcal{S}^+| = |\mathcal{S}_k^-|$. The experience is repeated until we reach the number of classifiers wanted by the user. This method is a generalization of Chen et al. (2004) where they use this same process for random forests.

Balance Cascade The idea of cascade algorithms is very similar to boosting in the sense that we want the new learners to rather focus on misclassified instances. The principle is based on the fact that if $x \in \mathcal{S}$ is correctly classified by $h_{k-1}(x)$, then it is considered as redundant in $\mathcal{S}$ and so is discarded such that h_k only accesses $\mathcal{S} \setminus x$. In Balance Cascade algorithms, the idea is the same but instead of removing correctly classified examples from $\mathcal{S}$, it rather only eliminates the examples from the majority class $\mathcal{S}^-$ that are well classified. In other words, Balance Cascade algorithm aims at building different classifiers (not necessarily weak) over a balanced class distribution. Balance Cascade follows the same bagging schema as Easy Ensemble by taking into account the example removed: $\mathcal{S}^+ \cup \mathcal{S}_k^- \subset \mathcal{S}^- \setminus x^-$ such that $|\mathcal{S}^+| = |\mathcal{S}_k^-|$.

For both methods, in practice, the decision threshold τ_k for each classifier h_k is defined by the user. More particularly, in Balance Cascade, the objective is to have models with a very low false negative rate. The final prediction of the learned classifiers for both methods is the average of the outputs for all learners using their specific thresholds:

$$F_L^* = sign(\sum_{k=1}^{L} h_k - \sum_{k=1} \tau_k).$$

In practice, any classifier that outputs scores is suitable for the h_k. In (Liu et al., 2009), the authors use Adaboost such as to do a weighted average using the weighted linear combination of the weak learners from the boosting model.

AdaCost Well-known cost-sensitive boosting methods are built around Adaboost. In fact, many versions of Adaboost have been invented such that the cost is taken into consideration during the learning process. The straightforward approach in Eq 1.20 is one of them and is called AdaC2. The update of the sample weights becomes:

$$w_{i,k+1} = w_{i,k} e^{-y_i \alpha_k h_k} c_{y_i}.$$

Nikolaou et al. (2016) present a comparison between the standard Adaboost method and many different cost-sensitive boosting methods. The conclusion, similar to the one we give in Chapter 2, is that using the standard Adaboost with calibrated probability estimates and shifted decision thresholds is the best option. Nonetheless, in specific applications such as fraud detection, these methods may be a solution to maximize the savings, for example. In Chapter 2 we show how this could be used with its advantages and drawbacks.

Why using Gradient Boosting in this thesis?

In Figure 1.12, we illustrate the re-weighted distribution by a boosting algorithm along its iterations. We take a simple imbalanced dataset (Figure 1.11) and learn a gradient boosting model. At each iteration, we can have the relative importance of each example "through the eyes" of the boosting algorithm by simply getting $|r_i^k| \forall i \in \{1, 2, ..., M\}$. Also, note that it is common practice to initialize the first weak learner regarding the imbalance ratio such that $\sum_{i=1}^{P} h_0(x_i^+) = 1$ and $\sum_{i=1}^{N} h_0(x_i^-) = -1$. Here we rather initialize $h_0 = 0$ and let the gradient boosting re-weigh the examples naturally.

We emphasize that, by nature, boosting algorithms focus on hard examples. When the class distribution is highly imbalanced, the boosting algorithm is driven by the minority class. In Figure 1.12, we show this phenomenon. After very few iterations ($T < 5$), the importance of a positive example far exceeds the importance of a negative example. Moreover, after more iterations, the only examples considered in the learning process are the ones near the positive examples. At that point, the weak learners find rules that only concern this little subset of points. This can be seen as an undersampling scenario where only a small fraction of negative points remains.

This little experiment motivates our choice of using boosting in the imbalance class setting. Indeed, boosting does not suffer from the class imbalance as much as other

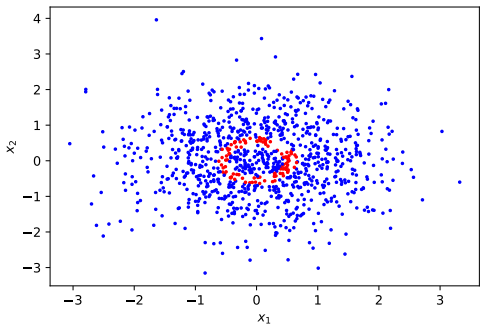

Figure 1.11: Imbalanced toy dataset. The blue points are part of the majority class while the red points are part of the minority class.

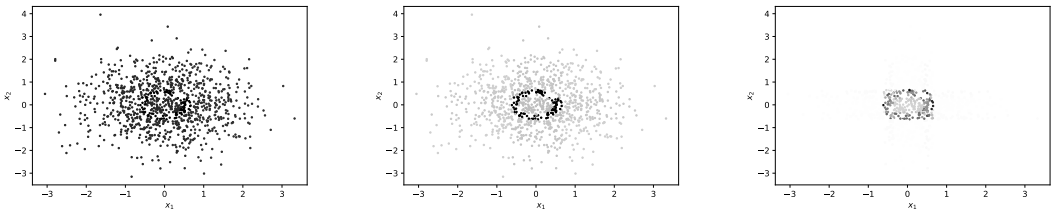

Figure 1.12: The three figure above shows how important a data point is in the eyes of boosting. The more intense to the black colour, the more important is an example at this boosting iteration. At first, it does not have a particular focus and sees all the examples equally important (most left figure). After very few iterations, a positive example has much more importance than a negative example (middle figure). Finally, the figure on the right shows the data point importance after many rounds. In fact, at this point, the new weak learner is learning over a very small subset containing the hardest samples to classify in the dataset.

learning algorithms. Although sampling methods combined with boosting might have a relevant impact in case of specific objectives, boosting naturally modifies the original class distribution by re-weighting iteratively the examples in regards to how well they were classified by the already learned weak models.

In the next chapter, we will focus on a specific problem where the class of interest is highly under-represented given by an industrial context in which this thesis takes place. In this setting, we will present imbalanced learning methods applied to a large scale dataset on credit card transactions.

Chapter 2

Learning with Extreme Imbalanced Data: Application to fraud detection.

Abstract

In this chapter, we first present a general overview of a specific class-imbalance problem: the anomaly detection. We then focus on the supervised fraud detection case accompanied with a brief description of Worldline's fraud detection system that served as a realistic environment for the realization of this thesis. This mainly comes down to dealing with a high imbalance setting coupled with large-scale issues that we characterize as extreme imbalanced data. In this context, we provide an analysis of the main metrics used for model evaluation and carry an experimental study with the state of the art method for fraud detection. Our analysis illustrates that the de facto standard machine learning techniques do not necessarily allow a behaviour adapted to the fraud detection problem we consider in the context of two settings: (i) sampling methods (ii) cost-based classification methods. We finish this chapter with a conclusion and open new directions for the extreme class imbalance data problem.

2.1 Introduction

In Chapter 1, we reminded some important notions and state of the art methods for class-imbalance problems. In this chapter, we extend the setting of imbalanced learning to a more extreme case. Specifically, we consider two main assumptions:

1. An extremely imbalanced dataset ($\rho > 500$)

2. A high number of observations ($M > 1,000,000$)

These settings arise recently in different fields such as bioinformatics (Triguero et al., 2015) or fraud detection (Wei et al., 2013) where companies have to deal with large-scale data. However, apart from few publications, it seems that there is a clear lack of study on datasets that meet the above conditions in the literature (Krawczyk, 2016). We define this as *extreme imbalanced data* where the imbalance ratio is extreme but the examples from the minority class are abundant.

An application that often fits the above conditions is the anomaly detection problem. More specifically, at Worldline, we focus on one of its subdomains which is the fraud detection task. In this chapter, we wish to unravel the effectiveness of previous imbalanced learning methods described in Chapter 1 within the fraud detection application. To this end, this chapter is divided in three main parts. First, we introduce the anomaly detection problem and one of its sub-domains, the fraud detection task. We then present the specificity of Worldline Fraud Detection System and the data. Finally, we discuss the cost sensitive approach for such problem and propose an experimental study of the state of the art methods for credit card fraud detection datasets.

2.2 Anomaly detection

Anomalies refer to the case where relatively few observations out of large amount of data are abnormal in the sense that they do not follow a well-defined notion of normal behaviour. Anomaly detection is a very active research topic (Chandola et al., 2009; Aggarwal, 2015; Akoglu et al., 2015; Ahmed et al., 2016). In the following we present different characteristics of these anomalies from a machine learning point of view.

The anomaly detection problem can be divided in two main settings that depend on whether labels are available or not. In the first case, supervised machine learning tend to be used while for the second case, unsupervised machine learning is the default choice. In both cases, while the class imbalance problem is present and can potentially make the learning process challenging, it turns out that it is not necessarily an issue in itself but is rather relative to the problem complexity. Figure 2.2 presents two different anomaly detection problems that are conducive to outlier detection (unsupervised learning). In this problem, it is rather easy for an unsupervised method to actually separate the normal data (in blue) from the

abnormal ones (in red). However, this is obviously not the case of most anomaly detection problems since 1) datasets can be very noisy and 2) some applications such as fraud detection are subject to concepts drifts that make the fraudulent behaviour hard to differentiate from the normal ones. This latter makes the task more complex even from a human expert point of view. Figure 2.3 is an example of datasets where anomalies would be indistinguishable from the normal examples if we did not have the labels. In this kind of setting, finding the anomaly is very hard for an unsupervised learning approach while it is very challenging but a more achievable task for a supervised learning approach. In our case, labels are available which allows us to use supervised learning approaches.

In general, we can distinguish three main types of anomalies that we summarize here:

Point anomalies A single data point is sufficient to identify its abnormality compared to normal observations. A simple example is shown on the left in Figure 2.2.

Contextual Anomalies In this type of anomaly, one can only spot the abnormal data point by looking at the context in which this observation belongs. For example, on the right of Figure 2.2 we see a time series problem where at some point the data do not follow the sinusoidal function. However, unless you take a step back and look at the behaviour of the points near the anomaly, there is nothing that tells you that this is indeed an anomaly simply (*i.e.* very similar observation are no anomalies at specific times).

Collective anomalies While the point anomaly case let us see that an example is actually an abnormal just by looking at this observation, a collective anomaly exists only if we can annotate an anomaly based on multiple instances. Figures 2.3 shows such a case where only by looking at all anomalies we can extract a pattern.

In fact, it is common to see all types of anomalies gathered in one complex problem. There exist many applications that lies in the anomaly detection problem. In *health care* an abnormal pattern can indicate a potential illness. In Zacharaki et al. (2009), the authors present an example of such application where they focus on tumour detection in MRI images. While this subject is very interesting, it suffers from a main difficulty which is to gather enough observations (MRI images) with their labels (expert decisions). In fact we can observe that most of the publications on this domain suffer from a lack of available data (Kourou et al., 2015) due to

their sensitive nature. While few recent studies are able to access large amount of real life data for building health care machine learning model (Chen et al., 2017), it still remains a rare commodity in the public domain. This application tends to belong in the point anomaly case where a single image is enough to detect whether a patient is ill or not. However there might be cases where having a prior knowledge on the patient can be relevant for the anomaly detection and thus considering the contextual anomaly case is relevant too.

Another promising domain for anomaly detection is in *security*. The main difference with health care is the large amount of data available and continuously increasing. Indeed, whether in sensors anomaly detection (Xie et al., 2011; Hill and Minsker, 2010), in Network intrusion detection (Tsai et al., 2009) or in Danger detection in crowded scene (Li et al., 2014b), the data is often available, however, having the ground truth is a different story. Indeed, the labels for such data are often not available for the simple reason that it is very costly for an expert to label such dataset (*i.e.* a single human is particularly slow for such task). In this application, many scientific contributions apply unsupervised or semi-supervised learning method but as it has been mentioned in Sommer and Paxson (2010), these methods still fail in real world systems where they suffer from specific constraints given by the environment. Recent studies as in Javaid et al. (2016) show promising results in a real-world application using deep learning methods, however, it assumes a labelled dataset. This domain most likely lies in the collective anomaly case where a single data point is very difficult to classify as an anomaly (wrong value of a sensor, network attacks). Therefore we rather look at a collection of observations to conclude that they are anomalies.

Fault diagnosis (Gao et al., 2015; Cai et al., 2017) is a growing application where the goal is to detect and identify abnormalities and faults as early as possible for minimizing performance degradation and avoiding dangerous situations. In Ince et al. (2016) they propose an approach based on neural networks to early motor fault detection. In this application, obtaining the real label can be challenging as one needs to observe real system failure to actually have the anomaly in the dataset. However, in this paper, the authors simulated the failures with simple tricks. While it can be argued that real failure might be very different from simulated ones, it allows the authors to completely control the dataset over which the machine learning model is trained. This application enters in the contextual anomaly case since this is mostly a time series problem.

The last domain of anomaly detection presented in this manuscript concerns *finance*. Naturally, financial machine learning applications are of great interest for

industrials. The most famous one is *fraud detection* (Bolton and Hand, 2002; Abdallah et al., 2016). Frauds have been observed since the first times of humanity and as long as they were done in a society there has always been a consensus to fight them. Machine learning research also increased through the years on this topic and is today a very active subject (Figure 2.1 presents the trend of research publications concerning the problem of fraud detection). Beside the attractive financial aspect of fraud detection, a reason for its popularity is also the number of applications that fits in this context:

- Click in mobile advertisement (Badhe, 2017).
- Taxes (Bonchi et al., 1999; Van Vlasselaer et al., 2016).
- Telecommunication (Farvaresh and Sepehri, 2011; Jain, 2017).
- Health insurance (Kirlidog and Asuk, 2012; Rawte and Anuradha, 2015).
- Automobile insurance (Wang and Xu, 2018).
- Check (Hines and Youssef, 2018) also subjects to many patents (Kotovich and Nepomniachtchi, 2007; Carney, 2001).
- Ratings/Reviews (Hooi et al., 2016).
- Credit card transactions (Bolton and Hand, 2002; Dal Pozzolo, 2015).

The fraud detection application is likely to lie in the contextual anomaly. For example, in case of transaction fraud detection, genuine pattern can be specific to the cardholder or to the merchant. That being said, collective anomalies (e.g. repeated transactions/ratings/clicks) and point anomalies (e.g. illogical characteristics of a data point) also appear in such data and allows us to detect the fraud.

In all these applications, a redundant gap between the public research and the industry can be found. In Ngai et al. (2011) the authors stress out the need for the industry to encourage research on real life systems since most developed research methods suffer from a lack of knowledge of industrial fraud detection systems. This problem was observed in the early work of Phua et al. (2004) on imbalanced learning applied to fraud detection. We come back to this matter in the next section.

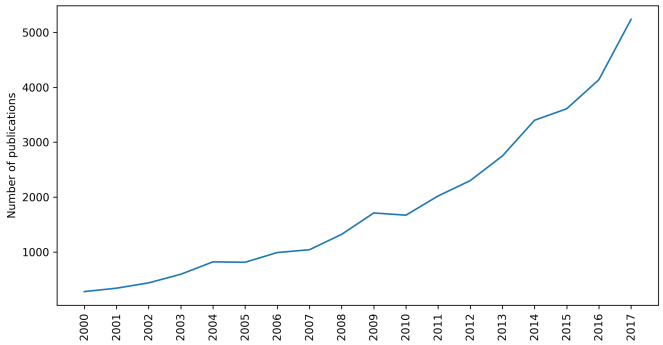

Figure 2.1: Number of publications on fraud detection with machine learning.

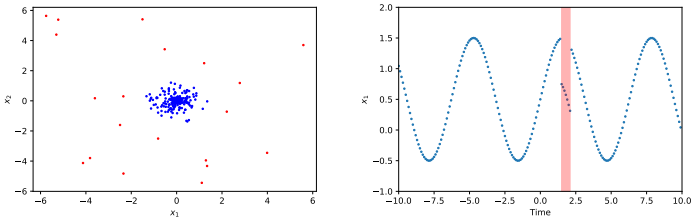

Figure 2.2: On the left, a typical dataset with a cluster in the middle and noisy example around. This can be solved with simple outlier detection algorithms. On the right, a time series problem. Both problems can be relatively easily solved with unsupervised learning algorithms.

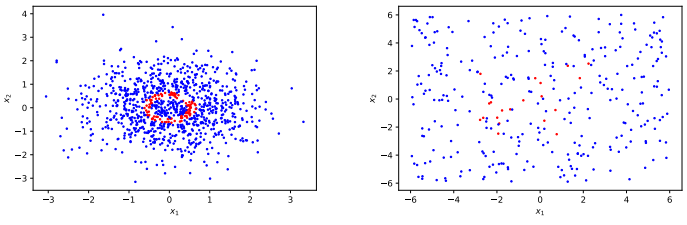

Figure 2.3: On the left, anomalies form a circle hidden inside the normal examples. On the right anomalies there is no direct pattern and detecting them is a challenge. In both problems, unsupervised learning is not an option. In this context, it is crucial to have labelled examples.

2.3 Credit Card Fraud Detection

In transactions, a fraud can be defined as stealing someone's identity to briefly acquire its privileges which has the main consequence for the fraudster to earn money illegally. There exist many different strategies for the fraudster to operate. We present briefly the most common procedures both in offline and online scenarios.

Offline fraud is a rare type of fraud where the fraudster actually steals the physical credit card from the cardholder or copy the magnetic stripe. In most cases, the victim contacts the bank to instantly block the lost credit card such that following transactions made on this card get refused.

Online fraud is today the typical fraud where fraudster steal the credit card's information through malicious online means. This type of fraud is the most dangerous as the fraudster is not aware of the theft (Patidar et al., 2011) and may happen through different strategies. Among the most frequent strategies, the first one is *skimming* where the idea is to steal the cardholder information during a genuine transaction through a modified payment terminal that stores all the card information. In recent years, ATMs have been used by skimmers to extract card information (Krebs, 2010) (i.e. by setting fake keyboard, a camera or other tools that capture the relevant card information). Another very popular strategy is called *phishing* where the fraudster uses a website to steal the card information by either cloning an existing one or simply by creating one with unsafe payment processes. Other strategies exist such as spreading Trojan-type malware which has become a very common practice. This kind of strategy is, however, not as effective as the previous ones given that it is hard to extract the relevant credit card information. Finally, simple tricks such as fooling the cardholder with malicious mails are also common.

The two types of fraud just described are still relevant today, however, it has not always been the case. Indeed, the credit card system was popularized in 1950 and, at that time, the first frauds were obviously offline. The first attempt to counter these frauds was to use a hologram that could be recognized by the merchant to prevent fake id cards (Lopata, 1987). When e-commerce started in the 90s, breaches were abundant for fraudsters to start designing fraudulent strategies. As soon as the online business spread all over the world, the need to have more elaborated fraud detection systems was crucial and first research papers on credit card fraud detection appeared with *expert systems* (Leonard, 1993). In these

systems, the fraud detection is mainly based on human experts that analyse the data thoroughly.

In the following we briefly present the state of the art of machine learning applied to the credit fraud detection problem. As we mentioned previously, this can be seen as a temporal problem where transactions from a card holder follow a certain order. Basically, a cardholder makes series of transactions that implicitly define its behaviour. Clearly, taking time into account is important and apart from models that can naturally use temporal relations, a common practice is to build new features that take these relations into account.

In Whitrow et al. (2009) and Bahnsen et al. (2016) the authors explore this idea and design new handful sets of features such as *average amount last 5 transactions*, *time since last transaction* ... In fact they also define new features to describe the cardholder behaviour such as the *average daily/week/month expenses*, *min and max amount spent in one transaction*. This approach is called feature engineering and is a common way to give more relevant information to the model. This is a common practice to quickly improve the performance of a model especially in fraud detection where the raw set of feature is not accompanied with cardholder historical information (Dal Pozzolo et al., 2014). As it turns out, terminals and merchants can be used in the exact same way to build historical features of a specific merchant or terminal (Van Vlasselaer et al., 2015). Finally, new methods show interesting results in building these features automatically and implicitly (Fu et al., 2016; Roy et al., 2018; Jurgovsky et al., 2018).

The creation of these new features is also a way to counter the *concept drift* that occurs through time. Indeed, fraudsters strategies tend to evolve with time. However, these new concepts are very hard to identify since they may be due to many changes. For example, during the Christmas period, each year, the behaviour of millions of customers change which makes the change in the fraudsters behaviour very hard to detect. One way of identifying them is often to observe a drop in the model performance, however, at this point it is often too late to recover from the loss. In Dal Pozzolo et al. (2015a), the authors present a method to build more relevant machine learning models using delayed feedback (i.e. labels arrive only a short period after the related transaction rather than being directly available). In this case, the authors take the time into account and thus relearn models as data arrive. This type of learning is called *incremental learning*. Typically, models a trained over some specific time periods (e.g. days, months,...) and retrained from scratch whenever sufficient data is available. In Kulkarni and Ade (2016), the authors use ensemble methods and imbalanced learning methods coupled with incremental learning.

Finally, as we may have implied previously, this task of fraud detection was found to be closely related to the domain of imbalanced learning He and Garcia (2008); Phua et al. (2004). In Chan et al. (1999), the authors introduce a first experimental study where they use AdaCost. Their work showed the great potential of ensemble methods on the credit card fraud detection task. However, they also emphasize on the fact that the costs are really hard to find and that the only solution to find them is through a lot of trials and errors. In Akbani et al. (2004), the authors propose a new algorithm based on SVM to compete with SMOTE on the specific case of fraud data. In fact, this imbalanced learning approach to credit card fraud detection was adopted by many recent contributions. In Padmaja et al. (2007), the authors use different sampling techniques. In Dal Pozzolo et al. (2014), the authors present different aspects of the credit card fraud detection problem from an interesting practitioner point of view and have a large experimental section dedicated to imbalanced learning methods. Lastly, many different papers, approach the problem with ensemble learning methods (bagging) (Zareapoor et al., 2015; Dal Pozzolo et al., 2013, 2018).

To summarize, we reviewed 3 main approaches to solve the fraud detection task on credit card transactions. The first one is manual feature engineering that relies on expert knowledge to handcraft new sets of features. This approach requires a lot of human effort to analysis the data thoroughly and may be unrealistic in case of concept drift where relevant features can change through time.

To cope with the drawbacks brought by human expertise, people started investigating automatic feature engineering. These techniques are mainly over models that handle spatial information (CNN or LSTM type of neural networks). In our context, this spatial information is *time* where these models try to extract patterns with regards to the sequence of transactions. These approaches seem to be promising, however, due to the potentially infinite amount of information that could be created as we look further in the past, it remains complicated to reach expert-level performance.

The first two approaches are data specific. In fact, the third approach that we identified is not an alternative to the two previous but could rather be combined with the two previous to build a more relevant model on such data. This is the imbalanced learning approach. The idea is to view the credit card fraud detection as an imbalanced classification problem. Typically, sampling methods and cost-sensitive learning methods are used. As it turns out, cost-sensitive learning comes in with some difficulties related to the credit card fraud detection that we detail in the following.

		Actual class	
		Fraud	Genuine
Predicted class	Fraud	c_a	c_a
	Genuine	Amt_i	0

Table 2.1: Cost matrix

2.4 Constraints of Cost-Sensitive Learning with Financial Cost

A approach that seems natural for fraud detection is to use cost-sensitive using financial information Sahin et al. (2013); Bahnsen et al. (2016). The idea is to apply specific costs for each transaction. In such problem we can write the confusion matrix as shown in Table 2.4 where c_a (regardless of its true label) is the cost of blocking a card and Amt_i the amount of the transaction x_i. The value of c_a is very specific to the fraud detection system and the agreement that Worldline has with banks and merchants. In our context setting a price for blocking on a specific transaction is very difficult since transactions follow different constraints (*e.g.* different merchants and different banks that not not always undergo the same process).

From an expert point of view, emphasizing on high amount transactions is not necessarily optimal to save money. This is a very counter-intuitive statement that we try to explain in the following.

The first important point is that frauds with low amounts are often a strategy used by fraudsters to test whether a credit card actually works. Often, these transactions are made in specific merchants that accept transactions with few or no security level. Moreover, when these low amount fraudulent transactions happen, it often announces bigger amount fraudulent transactions. We remind that machine learning models work in near-real-time thus they are not allowed to block the transaction being analyzed. Thus, blocking the card after the low amount fraudulent transaction is made is much more valuable than waiting for the high amount transaction. In other words, high amount fraudulent transactions would have to be accepted and customers refunded even if our model is good at detecting them.

A concerning point regarding the cost of a false negative (a fraud not detected by the system) is that our dataset comprises a lot of transactions where $Amt_i = 0$ (around 3% of the genuine transactions and around 12% for fraudulent transactions) which would imply that they are irrelevant to the model. The reasons for

such transaction to happen are many, for example, gas stations often charge a 0 euro transaction to make sure that the cardholder can pay the gas that he will take.

It follows from the previous points that setting a higher importance for transaction with high amounts would tend to have a negative impact on the recall (less fraud detected). Moreover, taking another approach such as generally increasing the weights for positive examples would tend to make the model produce more false alert. This could cause a negative impact on the precision.

To conclude on cost-sensitive learning with financial costs, while it seems like a good approach for the credit card fraud detection problem we presented some concerning points when applying such technique to the real use-case. That being said, there exist fraud detection problems where this is applied quite effectively such as in cheque fraud detection (Metzler et al., 2018) or financial statement fraud detection (Kim et al., 2016) while the choice of the cost c_a remains mysterious. That being said, we believe that further study with the production team of the FDS at Worldline to define these costs could potentially open different nice perspectives. Today, it remains very complicated to estimates the cost fairly.

2.5 Worldline's Fraud Detection System

Fraud Detection Systems can be complicated due to the real life constraints. We explain the one implemented at Worldline based on Figure 2.4 that was kindly shared by the authors of Dal Pozzolo et al. (2018). At the input of such system are the transactions coming from real-life events (we give more details on these transactions in the following section). The first step is a very basic step made at the terminal level where simple verification process are performed such as if the correct pin code was entered or if the account has enough money for the purchase. The transaction then enters the *transaction blocking rules* block.

What we just described enters in the **real-time** process where there is a strict response time limit ($\leq 10ms$). This process is able to block the transactions. Thus, the rules that make up this process must be very fast. An example of such rule is given in the following:

if $trx\_amount > 5 \times mean\_carholder\_amount\_spent$
and $trx\_country \neq cardholder\_country$
and $is\_ecom = False$ **then**

$$is\_fraud = True$$

In the **near-real-time** process, the main goal is to make a deeper analysis of the transaction. However, the main difference with the real-time process is that the transaction is never denied since it undergoes several operations that may take up to several minutes. They are composed of two main blocks: the expert-driven rules and the data-driven rules. In the former, experts (also called investigators) do a day-to-day analysis to build, update, and remove rules such that the performance remains stable. In reality, these rules are not trivial to compute. In fact, a single rule often contains several dozen lines of SQL code which obviously is complicated to maintain. These rules are monitored with specific metrics such as fraud detection rate (the true positive rate also called recall) and false alert rate (or false positive rate) and are removed if they exceed a certain threshold for any of the metrics. The experts are in charge to label the transactions either as fraudulent or genuine. This labeling part follows some guidelines. First, if there is no claim for a transaction to be a fraud after a given number of days (*i.e.* 30 days), confidence in the genuine label is close to 100% and is set in this way in the database. At the same time, experts have to check risky transactions raised by both the scoring rules and the data-driven model. If a transaction if found to be fraudulent, the card is eventually blocked and will be refused in future transactions. Finally, they are also in charge to label the transactions as fraudulent whenever a customer claims that a transaction in its bank account is a fraud.

Clearly, experts have to handle a lot of tasks which has a non-negligible cost. Specifically, Worldline can't hire enough experts to review all transactions since this would cost more than the frauds themselves and they are only able to achieve a specific amount of work in a given time. This makes the credit card fraud detection very costly in both money and time. This latter brings us to the data-driven part where rules are built automatically (*e.g.* machine learning models) based on the data they receive. As for the experts rules, their performance is monitored throughout time. It turns out that, today, in production, there is more effort toward expert systems rather the data-driven models. There are multiple societal reasons for that such as a poor confidence of customers in AI approaches for their solutions or simply a fear of human workers to be double-crossed by the AI. An important reason is the fact that, at the time or writing, machine learning models did not prove to reach human level performance on the task. One objective of this PhD thesis is to show that machine learning is today able to really help the experts in their daily job.

Regarding our data, it should be added that, even if the transaction is denied

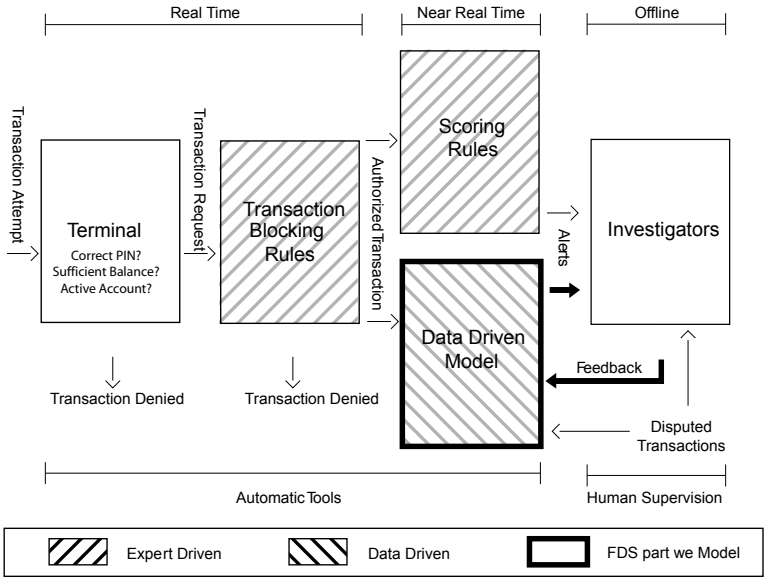

Figure 2.4: Fraud Detection System (FDS) at Worldline (Dal Pozzolo et al., 2018)

at **any** of verification process, it still enters Worldline's database. In the end, data are stored for a period of 6 months (with respect to the data protection laws) over which the experts can build new rules. After this period, data is removed permanently. In the following section we give a description of the data that Worldline receives continuously.

2.5.1 The Data

In this section we detail the data over which Worldline based its fraud detection system. First we would like to point out that a very small sample of Worldline's data containing $285,000$ transactions has been published on Kaggle [1] and is, at the time of writing, the most famous dataset on this platform with twice as much popularity score than the second most one [2]. This clearly confirms the attractiveness of the fraud detection task and should be a motivation for Worldline to open the data to the public domain (while making sure that they respect privacy and security).

In the data coming from real-life events and over which Worldline bases its rules, each transaction comes in the system with a handful set of features. It basically describes the transaction with different information such as the transaction date/amount, the cardholder birthday/gender/location, the card credit

[1] Most popular platform for data science competitions

[2] Link to the Kaggle dataset: `https://www.kaggle.com/mlg-ulb/creditcardfraud`

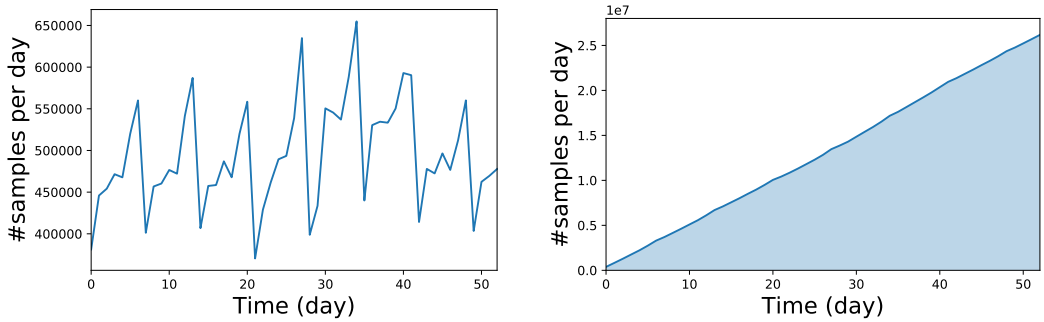

Figure 2.5: On the left the day-to-day volume of data arriving in the servers. On the right, the cumulative graph showing the amount of data gathered through time.

limit/type/expiration date and other different transaction specific variables. In this manuscript we do not focus on the feature engineering but rather take the dataset at hand. The data then have new information such as the average amount the cardholder spent for the past hour, past day, past week and other time-related features built on the same basis.

As we mentioned above, we want to study the class imbalance problem and in this sense we give different relevant figures in the following. In 2000, the percentage of fraud was estimated to occur once out of a thousand transactions ($1:1000$) (Lisboa et al., 2000). Today, the ratio of fraud is relatively similar with an average of one fraud for 700 transactions (1:700). However, the quantity of data increased with time. As a matter of fact, in 2013, Worldline received around $330,000$ transactions per day for a total of $120,000$ cards. Today, the average number of transactions received per day is around $680,000$ for a total of $440,000$ cards. In average per day, we have around 1100 fraudulent transactions for a total of 370 fraudulent cards.

It is interesting to note that today, approximately 90% of the frauds are made online (*i.e.* e-commerce). Figure 2.5 presents the day-to-day flow and the cumulative graph of data arriving in Worldline's FDS. In the left figure, we can distinguish days of the week by the number of transactions arriving in the FDS where Sundays are the lowest points in the repeated pattern and Saturdays are the highest peaks. In the right figure, we show that the quantity of data at hand grows quickly and linearly with time.

In machine learning it is rare to be able to visualize the data as the number of dimensions largely exceeds the number of dimensions that we can visualize. However, we believe that it may help to understand the task and thus we provide

different visualization using two different dimension-reduction methods known as Principal Component Analysis (Pearson, 1901) or PCA in the following and T-Distributed Stochastic Neighbour Embedding (Maaten and Hinton, 2008) or T-SNE in the following.

PCA seeks a linear combination of the features such as to extract the maximum variance. It then removes this variance, repeats the process with a second linear combination and repeat the process and so on. These linear combinations are called the principal components and are linearly uncorrelated to each other. Since we want to plot them in a 2 or 3 dimensional space, we only want to compute the first 2 or 3 principal components.

T-SNE is a more advanced method to visualize data in 2 or 3 dimensions. The main reason for using this method is its ability to compute non-linear combinations of the features which is not the case for the PCA. This offers a different visualization of the data. T-SNE first creates a probability distribution based on the distances between the examples. In a second step, it learns a low-dimensional space that follows this probability distribution as best as possible. Note that T-SNE has the main drawback of not defining a specific function which prevents the projection of new data point in the visual space.

Figure 2.6 presents the first two principal components ont he fraud dataset. It is rather clear that the centroid of fraudulent transactions is shifted compared to the centroid of genuine transactions. In Figure 2.7, we take advantage of T-SNE to plot the data with a non-linear transformation. Similarly to the PCA, T-SNE offers a nice visual interpretation where the fraudulent transactions appear in parts where the genuine transactions are less present. On this figure, we can distinguish sequences of transactions represented by clusters of points that follow each other in a sort of line. The left and right representations are taken over different periods.

Figure 2.8 shows a 3D visualization of the data built with T-SNE where we can see clusters that represent different behaviour. On the upper left, two clusters represent two different merchants. The one on the most left is a risky merchant in the sense that fraudsters use it to make fraudulent transactions. Interestingly, the frauds on the cluster are gathered on the same space and may be relatively easy to detect. These kinds of visualization may be very interesting from an expert point of view to understand the data, however, at the time of writing these tools still need improvement to allow an appropriate use for the experts. Moreover, as we stated, T-SNE do not allow new data points to be projected in the already built dimensional space which is a non-negligible drawback.

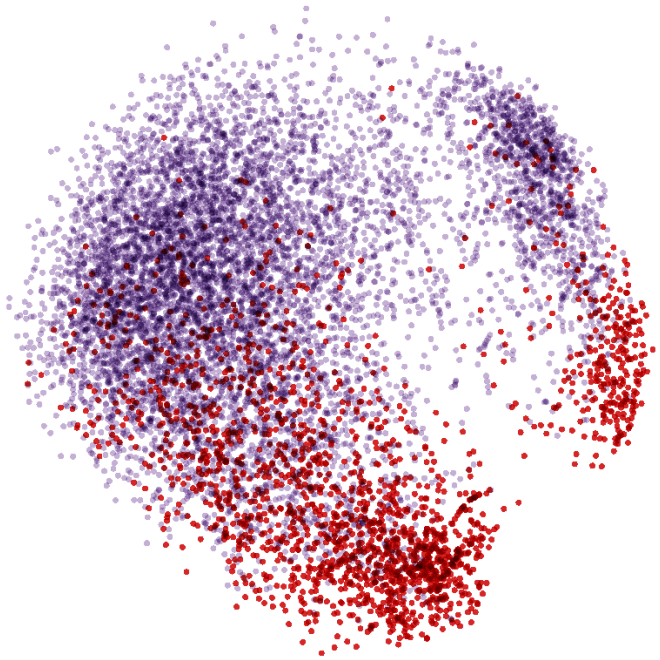

Figure 2.6: Representation with PCA in 2 dimensions of $5,000$ fraudulent transactions (in red) against $5,000$ genuine transactions (in blue).

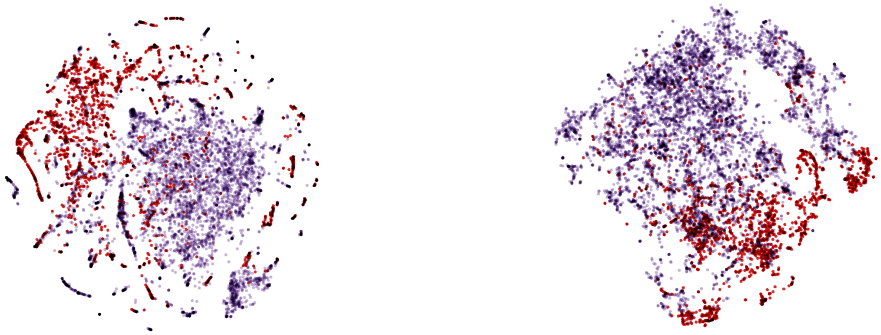

Figure 2.7: Representation with T-SNE in 2 dimensions of $5,000$ fraudulent transactions against $5,000$ genuine transactions. The two figures are taken over different periods.

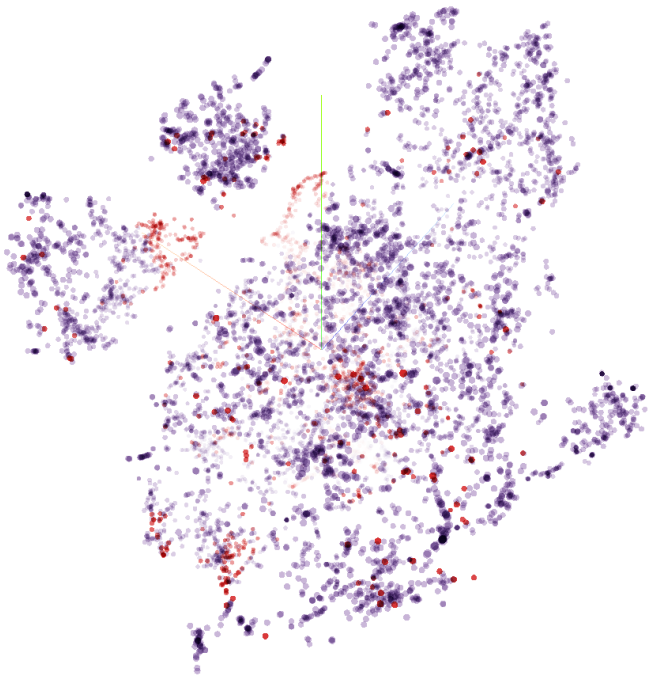

Figure 2.8: Representation with T-SNE in 3 dimensions of $5,000$ fraudulent transactions against $5,000$ genuine transactions.

It is important to note that these figures are highly misleading in the sense that the class distribution has been readjusted such as to ease the visualization. Indeed, to have an idea of the real class distribution, the number of blue points should actually be much times higher. In this setting, the genuine transactions (blue points) completely overlap the fraudulent transactions which makes the fraud detection problem a very difficult task. On the other hand, it also shows that sampling methods may be an interesting way to help the models to learn over this kind of data.

Lastly, as we mentioned previously, data are subject to changes in time (e.g. concept drift). This is also the case for the imbalance ratio that varies slightly with time as shown as Figure 2.9 where we see the positive ratio π change over time. The same goes for the time of the day (Figure 2.10. Note that the changes in this π can either be caused by fluctuations in the amount of genuine transactions or fraudulent transactions or both at the same time (*i.e.* it does not necessarily means more frauds).
In this section, we study the effectiveness of imbalanced learning methods for credit card fraud detection. The state of the art is today divided between three domains: cost-sensitive learning, sampling methods and ensemble methods. We present the in the following applied to the specific use-case.

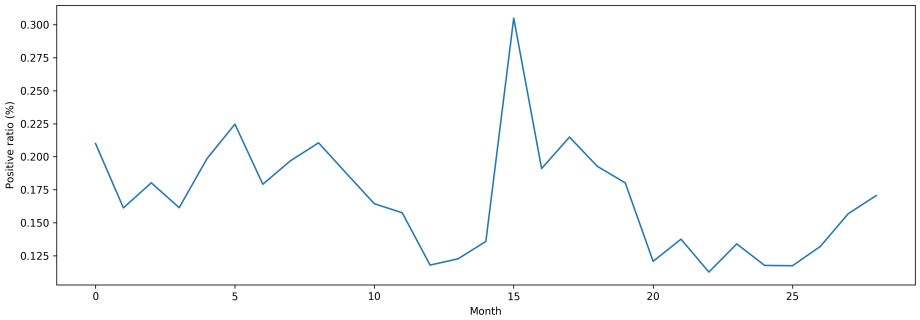

Figure 2.9: Positive ratio per month.

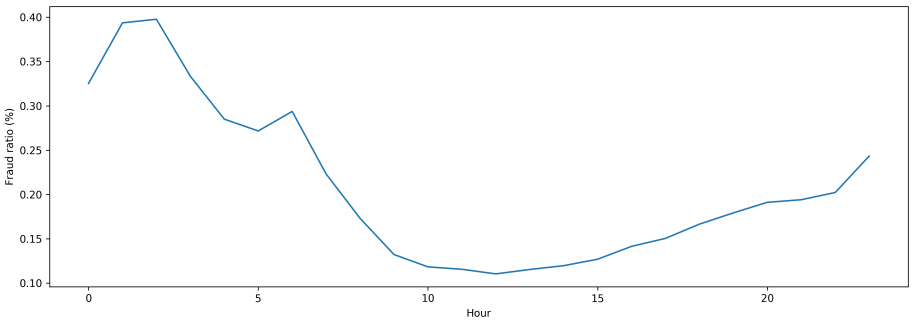

Figure 2.10: Positive ratio per hour.

2.6 Experiments with Imbalanced Learning Methods for Credit Card Fraud Detection

2.6.1 Experimental Setup

In this section we compare the most promising approaches to credit card fraud detection. In the previous we presented different studies that provide methods to tackle the credit card fraud detection problem (Dal Pozzolo et al., 2013, 2018; Zareapoor et al., 2015; Kulkarni and Ade, 2016). It turns out that, ensemble methods clearly outperform all other learning algorithm compared in these studies (*e.g.* SVM, Naive Bayes, K-NN). More specifically, three methods stand out:

1. Random undersampling with the Random Forest (RF).
2. SMOTE with RF.
3. EasyEnsemble with RF.

We remind briefly how these methods work. RF is a bagging method that combines different decision trees learned over different subsets. These trees are often not pruned thus with high variance. The specificity of RF compared to bagging lies in the random selection of features for each split of the trees. When coupled with undersampling or SMOTE, the training data over which the subsets are built is balanced such as to have a similar proportion of examples in all classes. While random undersampling remove examples from the majority class randomly, SMOTE, on the other hand, creates new synthetic examples based on the K Nearest Neighbour algorithm (see Section 1.4 for more details on these methods). Lastly, EasyEnsemble with RF relies on several balanced subsets created with random undersampling over which different RF will be trained (instead of a single one for random undersampling with RF).

Interestingly, gradient boosting (GB) hasn't been extensively used in such problem. In this experiment, we include GB as its internal properties allows a nice behaviour on imbalanced data (see Section 1.4.5). In Nikolaou et al. (2016) they support this idea by showing that boosting does not benefit from cost-sensitive learning since it naturally assigns higher cost for the minority class. We also combine GB with the three sampling methods, namely, random undersampling, SMOTE and EasyEnsemble.

An interesting property of RF and GB is their implementation using trees. The rules that make up these trees are interpretable and similar, in some sense, to

the expert rules which is a non-negligible asset. Indeed, interpretability is today a priceless feature for a machine learning model to be able to justify potential sensible decisions that the model could make.

We now present our experimental protocol with its specific settings. As we mentioned, the data we work on are related to time. Indeed, it would be unfair for a machine learning model to learn on the future to predict past frauds. The main reason is that it does not apply in the realistic case. Another important reason is that we observed that machine learning models could overfit in some way (by combining sets of features) to recognize the card and thus would have overoptimistic performance when tested over transactions close to the test regarding the time. To avoid that, we carefully split our in four parts: train, validation, gap and test in the chronological order. The gap is used such as to make sure we do not have the behaviour mentioned above. Figure 2.11 presents these splits.

In the following, we use 2 months of training data, 1 week of gap and 1 month for the test. The imbalance ratio is $\tau = 557$ or $\pi = 0.179\%$ where $\pi = \frac{P}{M} = \frac{1}{1+\rho}$ is the proportion of positive examples where $P = \sum_{i=1}^{M} \mathbb{1}(y_i = 1)$, $N = \sum_{i=1}^{M} \mathbb{1}(y_i \neq -1)$ and M is the total number of examples in the training set.

The experiments are carried over a machine with 800 gigabytes of RAM and 56 CPUs. In order to validate our models, we use the hold-out validation set and search for the best hyper-parameters for both the RF [3] and GB implementation [4] as follows:

- Gradient boosting hyper-parameters: `n_estimators` $\in \{10, 50, 100, 200, 300, 500, 1000\}$ that defines the total number of weak learners, `max_depth` $\in \{2, 5, 6, 10, 13, 15\}$ that stops the learning of the tree at a specific depth, `eta` $\in \{0.001, 0.01, 0.1, 0.3, 0.5\}$ the learning rate and `subsample` $\in \{0.4, 0.5, 0.6, 0.7, 0.8, 1\}$ that defines the proportion of examples randomly drawn over which a weak learner is trained. This last parameter also implements Friedman (2002) where they introduce the Stochastic Gradient Boosting algorithm.

- Random forest hyper-parameters: `n_estimators` $\in \{10, 50, 100, 200, 300, 500, 1000\}$ that defines the total number of trees in the ensemble and `max_features` $\in \{\sqrt{d}, \log_2(d)\}$ the number of features to consider at each split of the tree where d is the total number of features.

[3] Implementation from scikit-learn, `https://github.com/scikit-learn/scikit-learn`
[4] Implementation from Extreme Gradient Boosting, `https://github.com/dmlc/xgboost`

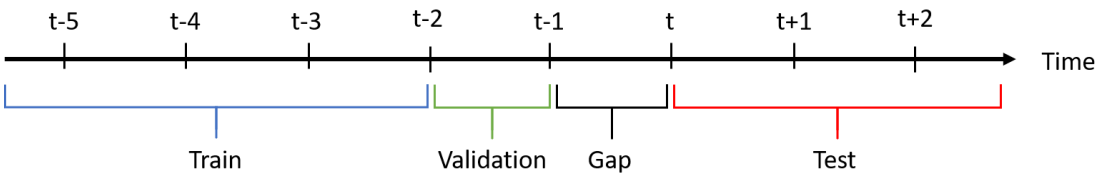

Figure 2.11: Experimental setup to validate and test a model on the credit card fraud detection dataset.

For the sampling methods, we selected different values for π_r, the proportion of positive examples in the balanced training set.

$$\pi_r \in \{0.002, 0.003, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$$

For each combination of hyper-parameters, we evaluate the model on the validation set with the metric over which we want to evaluate the models and select the set of hyper-parameters that maximizes it. The final reported results are then computed over the test set.

2.6.2 Side Effect of Sampling Methods

Now that we explained the experimental protocol we point out the fact that the sampling methods have some drawbacks that have been highlighted in the literature (Dal Pozzolo et al., 2015c,b):

1. It increases the variance of the classifier.

2. It produces posterior probabilities that may not be relevant on the test set.

In order to handle the variance, we take advantage of the nice generalization of ensemble methods and a large grid-search over a consequent hold-out validation set. The second effect mainly affects metrics based on the decision threshold. To illustrate this, we present the F_1 score for both RF and GB at different decision thresholds τ. Both model hyper-parameters are selected using the hold-out validation procedure.

Figures 2.12 and 2.14 present GB and RF respectively, trained over a balanced training set using undersampling while Figures 2.13 and 2.15 present the same models trained over the original training set. The goal of this experiment is to show how the posterior probability behave when training over the original and balanced training set. Note that, by default, the decision threshold is naturally set to $\tau = 0.5$.

A first observation is the clear difference between the training and test set figures (left vs. right figures). The first hypothesis why this happen is that the model could be overfitting the training set even with a hold-out validation. Another reason could be caused by a concept drift. Indeed, there might be concepts learned in the training set that do not appear in the test set.

Apart from their shape that heavily change from training to test sets, we can also see that they reach their maximum at different τ. When using undersampling (Figure 2.12 and 2.14), the optimal on the test set is very different from the one on the training set. This can be explained by the huge difference in the prior π in the two sets. For the models learned over the original training (Figure 2.13 and 2.15) it seems that the shape of the curves are more coherent between train and test. That being said, there exist a τ for which the RF, Figure 2.13, achieves a perfect F_1 score on the training set. This clearly shows that the model did overfit (as he test is far from being perfect. This is in fact not the case for GB, Figure 2.13, that show similar curves for both the training and test set and yet, the F_1 score is still not reached at the same τ.

A second observation can be made on the opposition between the balanced and original training set. Here we only focus on the performance obtained in the test sets (right figures). For both models, the conclusion is the same. The class distribution of the training set influence greatly the posterior probabilities. The probability $P(y = 1|x)$ given by the models is in fact shifted toward the class distribution. For example, in Figure 2.12 and 2.14, where the training set is balanced, the optimal threshold is near 1.0 while for the original dataset in Figures 2.13 and 2.15 the optimal threshold is close to 0.0.

From the previous observations, it is clear that one should carefully redefine the decision threshold. Another possibility is to find a new positive ratio achievable through undersampling such that the posterior probability becomes well calibrated. For example, Figure 2.16 presents a GB model trained over a re-sampled dataset with $\pi_r = 1.9\%$ or around 10 times higher than the original dataset. However, to obtain the right π_r that gives a good calibration of the posterior probabilities we had to experimentally test many different ratios which is very costly. Moreover, the best achievable F_1 score in this setting is lower than the one using the original training set.

It is clear that metrics such as the F_1 score are highly dependent on the decision threshold and that the default one is in often not optimal. For these reasons, in the following, we tune the decision threshold over the validation set such as to maximize the metric of interest (*i.e.* F_1 score).

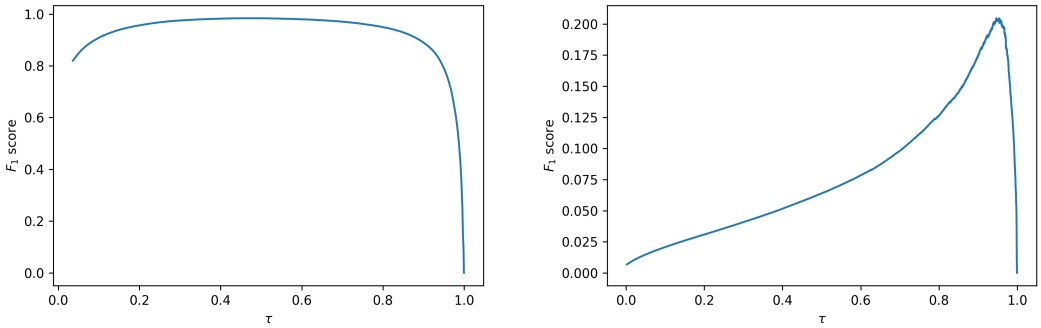

Figure 2.12: The figures present the F_1 score for the train (left figure) and test (right figure) at different decision thresholds. In this experiment undersampling was applied on the training set. This experiment was carried with a GB model.

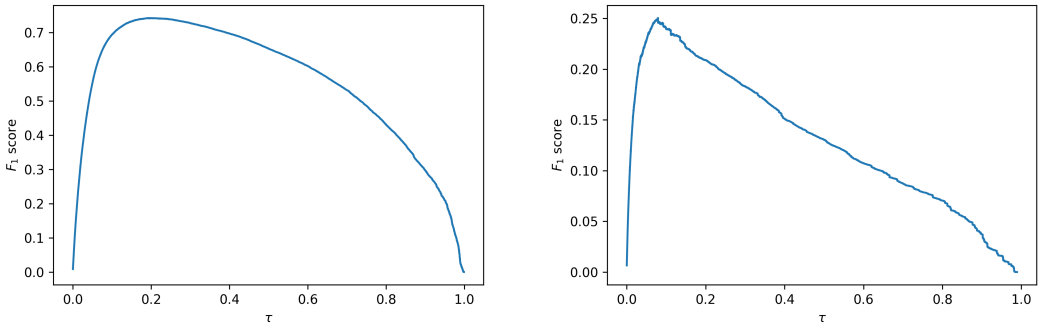

Figure 2.13: The figures present the F_1 score for the train (left figure) and test (right figure) at different decision thresholds. The training set was left in its original state. This experiment was carried with a GB model.

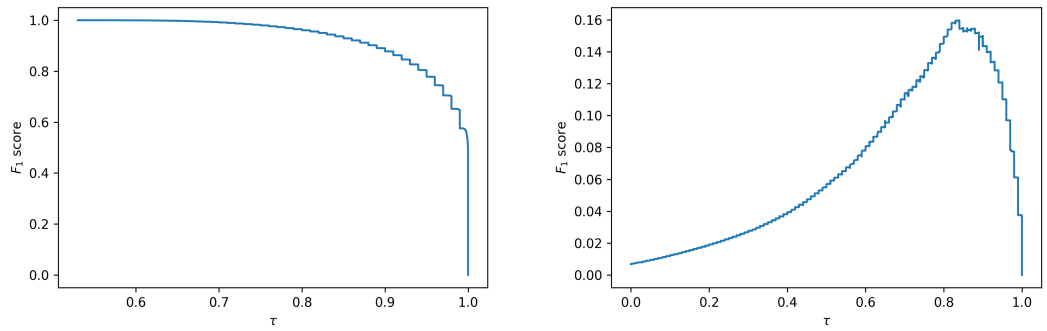

Figure 2.14: The figures present the F_1 score for the train (left figure) and test (right figure) at different decision thresholds. In this experiment undersampling was applied on the training set. This experiment was carried with a RF model.

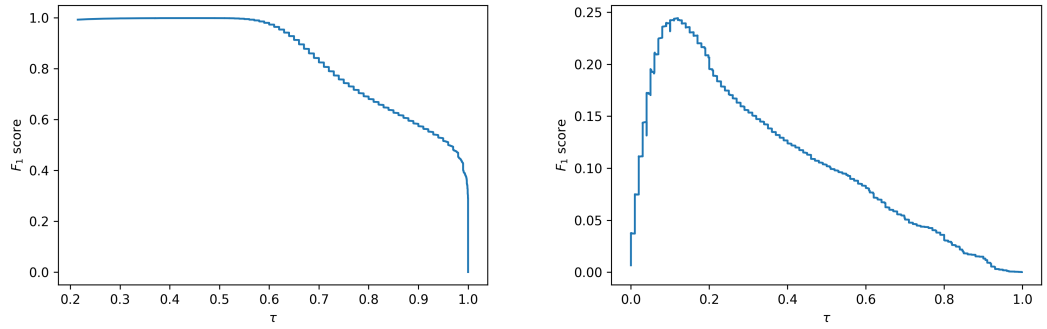

Figure 2.15: The figures present the F_1 score for the train (left figure) and test (right figure) at different decision thresholds. The training set was left in its original state. This experiment was carried with a RF model.

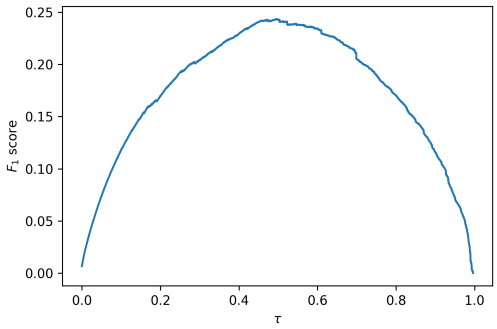

Figure 2.16: F_1 score at different decision threshold. The positive ratio was set at 1.9% for the training set using undersampling. This experiment was carried with a GB model.

2.6.3 Experiments

We compare RF and GB coupled with the sampling methods from above. We observe 3 different metrics that are very common in class imbalance problems, namely, AUCROC, AP and F_1 score. Since all methods are non-deterministic (*i.e.* all methods have a random variable) we average their results over 30 runs on the test set. The final results are reported in Table 2.2.

We quickly remind the intuition behind the 3 different metrics we use in this experiment. The F_1 score represents the harmonic mean at a specific decision threshold. It is quite clear that the lower the measure is the less precision **and** recall we can reach. Thus, when it decreases, the model does more false alerts while catching less frauds. The average precision (AP) is harder to interpret. This measure takes into account every possible relevant decision threshold and computes the precision for each of them. AP is finally the average of these precision. In other words, this measure evaluate the potential of the model or how well it does in average at every decision threshold. It also emphasizes in ranking well the positive instances. We come back to this in Chapter 3. Finally, the AUCROC has a very intuitive explanation as it is the probability of a randomly chosen positive example to be ranked above a randomly chosen negative example. In other words, an AUCROC of 0.5 gives a ranking of the examples completely random.

The first observation on the results is the optimal π_r that is in average very different for AUCROC compared to the other metrics. Indeed, while the F_1 score and AP are in agreement regarding this term, AUCROC prefers much higher π_r. To understand why this occurs we plot the three metrics while we chose different value of π_r with the random undersampling method in Figure 2.17. We can clearly see that the undersampling only decreases the model's performance in terms of both AP and F_1 score when $\pi_r > 0.004$. Interestingly, AUCPR and F_1 score find their maximum when $\pi_r = \pi + \epsilon$ where ϵ stands for a small positive quantity. It turns out that the proportion of positive examples on the test set is slightly higher ($\pi_{test} = 0.214$). Indeed, our models are built over a prior π_r that is assumed to be the same on the test set. For this reason, when $\pi_r \approx \pi_{test}$ both metrics find their maximum.

The behaviour of RF and GB are fairly different regarding the value of π_r. Indeed, for RF, undersampling seems to have a non-negligible positive effect on the AUCROC. In fact by increasing π_r from $\pi_r = 0.002$ to $\pi_r = 0.1$ AUCROC is greatly increased. This is in contrast with GB that remains somewhat stable regardless of the π_r values. In the end, results in Table 2.2 suggest that only EasyEnsemble

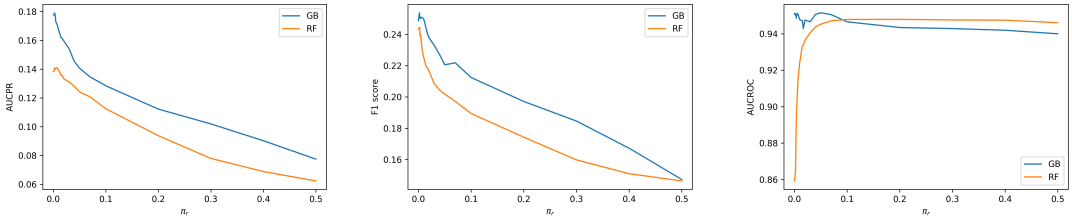

Figure 2.17: From left to right, AP, F_1 score and the AUCROC reported at different π_r using undersampling.

was able to reach the best results that were obtained by the RF and GB models without sampling methods.

It is interesting to note that the resulting RF model learned over the original training set is around $3,000$ megabytes in size while GB only is 10 megabytes. This can be explained by the trees generated in both methods. In RF, they are very deep since there is no pruning. However, for GB, the trees are "weak" (with a maximum depth of 15) which makes the model much lighter.
An interesting observation can be made by looking at the optimal π_r chosen by every sampling method. Indeed, $\pi_r \approx \pi$ for all of the three sampling methods. The dataset obtained with this π_r is in fact still very imbalanced. In this sense, sampling methods do not seem to have a strong impact on the model's performance. The decision threshold is, however, very important. Indeed, the F_1 score that would be obtained with the standard threshold $\tau = 0.5$ would be much worse. Figure 2.18 illustrates such phenomenon. In this figure, we use different values of π_r. In this case, the F_1 score computed is left by default which has the main consequence for the undersampling to increase performance of the models. As one may notice, this figure is very similar to Figures 2.15 and 2.13 (right figures) that are learned over the original distribution. It seems that the optimal decision threshold is very correlated to the sampling ratio π_r where the best performance of the models is achieved.

Regarding the metrics, they emphasize on different things. More specifically, AUCROC gives a different conclusion than the F_1 score and AP. Moreover, for RF, a perfectly balanced dataset offer almost the best performance in AUCROC while it gives the worst performance on the two other metrics. In our case, a lower AP means higher probability of making false positives and having less true positives (*i.e.* less precision, less recall) which is absolutely not desirable. AS it turns out, the F_1 score and AP are highly correlated. We provide a experimental study on their correlation in Appendix. A.

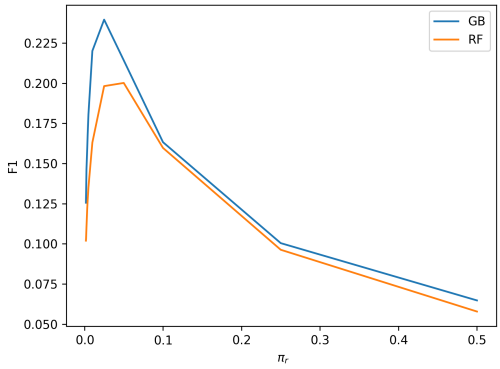

Figure 2.18: F_1 score reported for different positive ratio using undersampling without tuning the decision threshold.

This observation follows on Davis and Goadrich (2006) that presented this problem earlier on. Recently, in Saito and Rehmsmeier (2015), the authors presents a framework where AP is more informative than AUCROC. The main reason standing out why one would prefer to use AUCROC over AP is its ability to interpolate between points in the ROC curve while it is not possible to do the same in the PR curve. Another reason is the AUCROC being invariant on the prior π. That being said, a recent study presents a possible way to build the PR curve such that it can benefit from the same advantage as the ROC curve by redefining the precision and recall (Flach and Kull, 2015). In fact this problem arises when the data points are relatively far from each other in the precision and recall space. Indeed, interpolating between distant points in the PR space would highly overestimate the AP. In our case, when working over the fraud dataset the models we build give a sufficient number of scores to make the problem of interpolating between points negligible.

Today, a lot of different publications on class imbalanced problems still report AUCROC as the evaluation metric. More importantly, recent studies (Brabec and Machlica, 2018; Haixiang et al., 2017) rise a concern in the use of evaluation metrics for class imbalance problems. In fact, in Haixiang et al. (2017) the authors report that 38% of applied papers dealing with class imbalance problems in different domains use the accuracy as an evaluation metric. As we have presented in Section 1.4, using the accuracy in the class imbalance setting can be very misleading. Today, metrics for class imbalance learning should be carefully chosen such as to represent as best as possible the desired performances.

Table 2.2: Experiment results on the fraud dataset. The positive ratio chosen π_r is specified for each method where π is the original positive ratio.

Method	AUCROC (π_r)	F_1 score (π_r)	AP (π_r)
Undersampling + RF	0.9137 ± 0.0006 (0.500)	0.2417 ± 0.0118 (0.002)	0.1439 ± 0.0062 (0.002)
Undersampling + GB	$\mathbf{0.9291 \pm 0.0009}$ (0.020)	0.2458 ± 0.0033 (0.003)	0.1656 ± 0.0003 (0.003)
SMOTE + RF	0.8616 ± 0.0012 (0.500)	0.2293 ± 0.0212 (0.003)	0.1354 ± 0.0057 (0.003)
SMOTE + GB	0.9266 ± 0.0010 (0.002)	0.2296 ± 0.0199 (0.002)	0.1554 ± 0.0042 (0.002)
EasyEnsemble + RF	0.9193 ± 0.0002 (0.500)	0.2459 ± 0.0041 (0.002)	0.1580 ± 0.0036 (0.002)
EasyEnsemble + GB	$\mathbf{0.9292 \pm 0.0002}$ (0.050)	0.2465 ± 0.0038 (0.003)	$\mathbf{0.1684 \pm 0.0021}$ (0.003)
RF	0.9018 ± 0.0076 (π)	0.2464 ± 0.0051 (π)	0.1579 ± 0.0028 (π)
GB	$\mathbf{0.9291 \pm 0.0004}$ (π)	$\mathbf{0.2473 \pm 0.0027}$ (π)	$\mathbf{0.1682 \pm 0.0019}$ (π)

2.7 Conclusion

In this chapter, we covered the anomaly detection task and more specifically the fraud detection problem applied to credit card transactions. Fraud detection is today a very popular topic that has received a lot of attention from the machine learning community as it is a practical domain for supervised learning (*i.e.* it gathers both data and labels). We presented an overview of imbalanced learning methods applied to credit card fraud detection.

In a second step we reviewed the state of the art imbalanced learning method applied to fraud detection. We presented the cost-sensitive method and why this could be of great interest from a financial point of view (*i.e.* emphasizing on high amount fraudulent transactions). However, setting the costs is not trivial and we showed that focusing on high amount transactions is not always ideal.

It comes out that ensemble methods stand out because of their performance and generalization behaviour. In order to understand how these methods impact the model, we presented three different strategies that appeared as the best methods in the literature:

1. Random undersampling,
2. SMOTE,
3. EasyEnsemble.

All these methods were used with a Random Forest Gradient Boosting. Models were evaluated using three different metrics namely, AUCROC, AP and F_1 score.

We observed a different behaviour from AUCROC compared with the F_1 score and the AP. The former tends to be maximized as the training set becomes more balanced while the latter is clearly showing to be maximized as the training set

prior gets closer to the original one. Not only do they behave differently but they also completely disagree on the choice of the best model. This latter prompted us to investigate more on the meaning of these measures and we concluded that the AUCROC as well as the accuracy are both misleading for credit card fraud detection data or more generally extreme imbalanced data problems.

In the end, sampling and cost-sensitive are a lot effort for few or no reward. Indeed, SMOTE and its hybrid counterparts such as SMOTEBoost are very impractical in our context where millions of examples have to be generated. Undersampling however, benefits from easing the learning process of the models by removing a large set of genuine transactions. That being said, we did not notice a significant improvement at any imbalance ratio in terms of AP and F_1 score. Undersampling does help RF to increase the AUCROC at the cost of lowering the precision. We believe that the loss of information caused by undersampling our data is dramatically impacting the decision that becomes less precise and biased toward a different distribution. While AP and F_1 score are better metrics to measure the model's performance on such dataset, we showed that the F_1 score is biased toward the decision threshold.

To summarize, we identified two main problems in fraud detection. The first is the *extreme imbalanced data* setting. To tackle this problem, we build upon the observations made in this chapter and address the problem by neither using sampling nor cost-sensitive methods. In fact, we rather use the original training set. Moreover, we believe that guiding our model toward maximizing the metric of choice would allow us to have better performances. In our case, we choose to focus on AP for 2 main reasons: 1) It disregards the decision threshold which allows to express the entire potential of the model (regardless of the decision threshold). 2) It takes into account both precision and recall which are two very important measure in fraud detection. In the following chapter, we present a way to optimize this metric in a GB framework. The second identified problem is the uninterrupted flow of data entering the system. This problem clearly makes learning in the offline setting unrealistic as time passes. This latter will be our main focus in Chapter 4.

Chapter 3

Learning with imbalanced data from a learning to rank point of view

This chapter is based on the following publication

Frery, Jordan, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. "Efficient top rank optimization with gradient boosting for supervised anomaly detection." In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 20-35. Springer, Cham, 2017.

Abstract

In this chapter we address the anomaly detection problem in a supervised setting where positive examples might be very sparse. We tackle this task with a learning to rank strategy by optimizing a differentiable smoothed surrogate of the so-called *Average Precision* (AP). Despite its non-convexity, we show how to use it efficiently in a stochastic gradient boosting framework. We show that using AP is much better to optimize the top rank alerts than the state-of-the-art measures. We demonstrate on anomaly detection tasks that the interest of our method is even reinforced in highly imbalanced scenarios.

3.1 Introduction

As discussed in the first two chapters of this manuscript, there exist several methods to get rid of the issues due to imbalanced datasets. The most famous are sampling-based strategies, either by undersampling or oversampling the

data (Chawla et al., 2002; Ramentol et al., 2012). The former aims at removing instances from the majority class while the latter creates synthetic data from the minority class. Several hybrid methods such as SMOTEBoost (Chawla et al., 2003), RUSBoost (Seiffert et al., 2010) and Adacost (Fan et al., 1999) combine a learning algorithm with a sampling or cost-sensitive methods. However, it turns out that these approaches have been shown to be hard to use when facing highly imbalanced situations (Dal Pozzolo et al., 2015b) leading to either insufficient generated diversity (by oversampling) or too drastic reduction of the dataset size (by undersampling).vIn addition, sampling methods induce a bias in the posterior probabilities (Niculescu-Mizil and Caruana, 2005; Dal Pozzolo et al., 2015c). We discussed all this problems in Chapter 2.

On the other hand, it is worth noticing that a peculiarity of the use cases mentioned above is the need to resort to a (often limited) number of human experts to assess the potential anomalies found by the learned model. Actually, our contribution stands in a context where the number of false positives (FP) may be significantly larger than the false negative (FN) due to the high class imbalance and where the impact of FP is very penalizing. For example, in fraud detection for credit card transactions, it is out of the question to automatically block a credit card without the expert approval (which may risk the confidence of customers having their credit card falsely blocked). In this context, the goal of the automatic system is more to make the shortest list of alerts preventing the expert from going through thousands of transactions. In other words, one aims at maximizing the number of true positives in the top rank alerts (*i.e.* the so-called *precision*) rather than discriminating between abnormal and normal cases.

This is the reason why we tackle in this chapter the supervised anomaly detection task with a *learning to rank* approach. This strategy has gained a lot of interest in the information retrieval community (Liu, 2011). Given a query, the goal is to give the most relevant links to the user in a small set of top-ranked items. It turns out that apart the notion of query, the anomaly detection task can relate to this setting aiming at finding the anomalies with the highest precision without giving too many genuine examples to the experts.

In such settings, different machine learning algorithms have been efficiently used such as SVMs (*e.g.* SVM-Rank (Joachims, 2002), SVM-AP (Yue et al., 2007)) or ensemble methods (*e.g.* random forest (Breiman, 2001), boosting (Freund et al., 1999)). It turns out that gradient boosting has shown to be a powerful method on real life datasets to address learning to rank problems (Chapelle and Chang, 2011). Its popularity comes from two main features: (i) it performs the optimization in

function space (Friedman, 2001) (rather than in parameter space) which makes the use of custom loss functions much easier; (ii) boosting focuses step by step on difficult examples that gives a nice strategy to deal with imbalanced datasets by strengthening the impact of the positive class. In order to be efficient in learning to rank problems, gradient boosting needs to be fed with a loss function leading to a good precision in the top-ranked examples.

In the literature, many approaches resort to pairwise loss functions (Freund et al., 2003; Burges et al., 2005; Herschtal and Raskutti, 2004), typically checking that every positive example is ranked before any negative instance. Note that all those methods implicitly optimize the area under the ROC curve. Therefore they aim at minimizing the number of incorrectly ranked pairs but do not directly optimize the precision of top ranked items as shown in (Burges, 2010).

To overcome this issue, recent works in learning to rank suggested optimizing other criteria like the Average Precision (AP) or the Normalized Discounted Cumulative Gain (NDCG) such as in Adarank (Xu and Li, 2007), LambdaMART (Wu et al., 2010) or LambdaRank (Burges et al., 2007). It has been shown that both AP and $NDCG$ are much more suited for enhancing ranking methods. However, due to the non-convexity and non-differentiability of those criteria, the previous methods rather work on standard surrogate convex objective functions (such as the pairwise cross-entropy or the exponential loss) and take into account the AP and $NDCG$ in the form of weighting coefficients only. In other words, the gradients are not computed as derivatives of AP and $NDCG$. Therefore, used in this way, these criteria only tend to guide the optimization process in the right direction. We claim here that there is room for doing much better and directly considering the analytical expressions of those criteria in a gradient boosting method.

In this paper, our contribution is three-fold: (i) focusing on AP, we show how to optimize a loss function based on a surrogate of this criterion; (ii) unlike the state-of-the-art learning to rank methods requiring a quadratic complexity to minimize the ranking measures, we show that AP can be handled linearly in gradient boosting without penalizing the quality of the solution; (iii) compared to the state of the art, we show that our method allows us to highly improve the quality of the top-ranked items. We even show that this advantage is much larger when the imbalance of the datasets is very important. This is a particularly interesting feature when addressing anomaly detection problems where the positive examples are very sparse.

The rest of this paper is organized as follows : In Section 2 we first introduce our notations, then describe our performance measures and present an approximation

to AP. We then describe our method in a boosting framework and define a more suitable smoothed AP as the loss function in Section 3. We demonstrate the effectiveness of our work in the experiments section where we compare several state of the art machine learning models in Section 4.

3.2 Evaluation Criteria and Related Work

We remind some notations used in this chapter. We consider a binary supervised learning setting with a training set $\mathcal{S} = \{z_i = (x_i, y_i)\}_{i=1}^{M}$ composed of M labelled data, where $x_i \in \mathcal{X}$ is a feature vector and $y_i \in \{-1, 1\}$ is the label. In imbalanced scenarios, $y = 1$ often describes the minority (positive) class while $y = -1$ represents the majority (negative) class. Let P (resp. N) be the number of positive (resp. negative) examples such that $P + N = M$. We also define $\mathcal{S}^+ = \{z_i^+ = (x_i^+, y_i^+) | y_i = +1\}_{i=1}^{P}$ and $\mathcal{S}^- = \{z_i^- = (x_i^-, y_i^-) | y_i = -1\}_{i=1}^{N}$ where $\mathcal{S}^+ \cup \mathcal{S}^- = \mathcal{S}$. We assume that the training data $z_i = (x_i, y_i)$ is independently and identically distributed according to an unknown joint distribution $\mathcal{D}_{\mathcal{Z}}$ over $\mathcal{Z} = \mathcal{X} \times \{-1, 1\}$.

In this work, we aim at learning from S a function (or hypothesis) $f : \mathcal{X} \to \mathbb{R}$ that gives a real value to any new $x \in \mathcal{X}$.

As already dicussed in Chapter 1.1, assessing the quality of f in an imbalanced scenario requires the use of an appropriate evaluation criterion.

In Chapter 2 we showed that the different possibilities evoked in Section 1.4 often induce different unexpected effect that may reduce the potential performance of a model. In this part, we focus on the *learning to rank* scenario. Rather than discriminating examples belonging to the positive and negative classes, we rather aim at ranking the data with a maximal number of TP in the top ranked examples which can be interpreted as a short list of alerts.

This setting is actually very relevant for fraud detection systems relying on expert validation such as Worldline Fraud Fetection Fystem, since one expert may just have to check the top k instances reported in the list. In this context two measures are well used in the literature: the *pairwise* $AUCROC$ measure and the *listwise* average precision AP that we recall below.

From a statistical point of view, we remind that the $AUCROC$ represents the probability that a classifier ranks a randomly drawn positive instance higher than a randomly chosen negative one. The expression of this measure is equivalent to the Wilcoxon-Mann-Whitney statistic (Hanley and McNeil, 1982):

$$AUCROC = \frac{1}{PN}\sum_{i=1}^{P}\sum_{j=1}^{N} \mathrm{I}_{0.5}(f(x_i^+) - f(x_j^-)), \tag{3.1}$$

where $\mathrm{I}_{0.5}$, is a special indicator function that yields 1 if $f(x_i^+) - f(x_j^-) > 0$, 0.5 if $f(x_i^+) - f(x_j^-) = 0$ and 0 otherwise. In the following we will use the classic indicator function $\mathrm{I}(*)$ that yields 1 if $*$ is true, 0 otherwise.

$1 - AUCROC$ has been exploited in Rankboost algorithm (Freund et al., 2003) as an objective function where the authors use the exponential as a surrogate to the indicator function. Let $\ell_{roc}(z_i, f) = \frac{1}{N}\sum_{j=1}^{N} e^{(f(z_j^-) - f(z_i^+))}$ be the loss suffered by f at z_i. We get the following upper bound on $1 - AUCROC$:

$$1 - AUCROC \leq \frac{1}{P}\sum_{i=1}^{P}\frac{1}{N}\sum_{j=1}^{N} e^{(f(z_j^-) - f(z_i^+))} = \frac{1}{P}\sum_{i=1}^{P}\ell_{roc}(z_i, f) = \mathbb{E}_{z_i \in S^+}\ell_{roc}(z_i, f) \tag{3.2}$$

We can notice that this objective is a pairwise function inducing an algorithmic complexity $\mathcal{O}(PN)$. Moreover, as illustrated later in this section, earlier in Chapter 2 and shown in (Burges, 2010), ℓ_{roc} is not well suited to maximize the precision in the top ranked items.

A better strategy consists in using an alternative criterion based on the average precision AP that we presented in Equation 1.19. In fact, we can redefine this AP to a simpler form where we assume that our predictions are complete ranking (no ties).

$$AP = \frac{1}{P}\sum_{i=1}^{P} p(k_i), \tag{3.3}$$

where $p(k_i)$ is the precision with respect to the rank k_i of the i^{th} positive example. Since the rank depends on the outputs of the model f, we get:

$$p(k_i) = \frac{1}{k_i}\sum_{j=1}^{P} \mathrm{I}(f(x_i^+) \leq f(x_j^+)) \tag{3.4}$$

with

$$k_i = \sum_{j=1}^{M} \mathrm{I}(f(x_i^+) \leq f(x_j)). \tag{3.5}$$

Plugging Eq.(3.4) and Eq.(3.5) in Eq.(4.9) we get:

$$AP = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{\sum_{j=1}^{M}\mathrm{I}(f(x_i^+) \leq f(x_j))}\sum_{j=1}^{M}\mathrm{I}(y_j = 1)\mathrm{I}(f(x_i^+) \leq f(x_j^+)). \tag{3.6}$$

Figure 3.1: Two rankings (with two positives and eight negative examples) ordered from the highest score (at the top) to the lowest. On the left, we get $AUCROC = 0.63$ and $AP = 0.33$. On the right, $AUCROC = 0.56$ and $AP = 0.38$. Therefore, the two criteria disagree on the best ranked list.

AP has been used in recent papers to enhance learning to rank algorithms.

In Burges (2010); Burges et al. (2007), the authors introduce a new objective function, called LambdaRank, which can be used with different criteria, including AP. This function depends on the criterion of interest without requiring to compute the derivatives of that measure. This specificity allows them to bypass the issues due to the non differentiability of the criterion. The objective function takes the following form:

$$\frac{1}{N}\sum_{i=1}^{P}\ell_{\lambda Rank}(z_i^+, f) \tag{3.7}$$

with $\ell_{\lambda Rank}(z_i^+, f) = \frac{1}{N}\sum_{j=1}^{N}\log(1+e^{-(f(x_i^+)-f(x_j^-))})|AP_{ij}|$ the loss suffered by f at z_i. Here, $|AP_{ij}|$ is the absolute difference in AP when one swaps, in the ranking, example x_i with x_j. LambdaMART (Wu et al., 2010) made use of LambdaRank in a gradient boosting method and got good results as reported in (Chapelle and Chang, 2011). However, it is worth noticing that in this algorithm, the analytical expression of AP as defined in Eq.(3.6) is not involved in the calculation of the gradient. $|AP_{ij}|$ can be viewed as a weighting coefficient which hopefully tends to guide the optimization process towards a good solution. One objective of this chapter is to directly use AP in the algorithm and therefore to use the same criterion at both training and test time.
In Section 2.6, we highlighted some drawbacks of using $AUCROC$ for extreme imbalanced data. In the following, we present the effect of $AUCROC$ and AP in terms of quality of top ranked items. Figure 3.1 compares these criteria in two different situations according to the location of two positive (in dark colour)

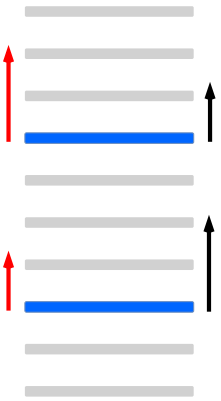

Figure 3.2: Comparison of the emphasis given by AP (arrows on the left) and the emphasis given $AUCROC$ (arrows on the right) (Burges, 2010). One can compare this emphasis to the intensity of gradients w.r.t the examples if AP and $AUCROC$ were continuous functions.

and eight negative (in light colour) examples that are ordered according to their predicted scores (highest score at the top). The key point of this figure is to show that $AUCROC$ and AP disagree on which list is the best. $AUCROC$ prefers the list on the left because the positive examples are rather well ranked even though the first three items are negative. Therefore, we can note that this criterion is very relevant if we are interested in classifying examples into two classes, for example, the classifier being based on a threshold (likely after the fifth rank, here) splitting the items into two parts. AP is in favour of the list on the right because it prefers to champion the top list accepting to pay the price to miss some positives. This criterion is thus very relevant to deal with anomaly and fraud detection where the goal is to provide the shortest list of alerts (here, typically the first two items) with the largest precision.

Figure 3.2 (inspired from Burges (2010)) illustrates graphically how the emphasis is done while computing gradients from pairwise loss function such as $AUCROC$ (black arrows on the right) or a listwise loss function such as AP (red arrows on the left) respectively. We can notice that a learning algorithm optimizing the $AUCROC$ would champion first the worst positive to get a good classifier (w.r.t. an appropriate threshold) while the AP would promote first the best positive to get a good top rank.
The previous analysis shows the advantage of optimizing AP in a learning to rank algorithm. This is the objective of the next section where we introduce a differentiable expression of AP in a gradient boosting algorithm.
The previous analysis shows the advantage of optimizing AP in a learning to rank algorithm. In the next section, we propose a method for optimizing AP in a

gradient boosting algorithm. Actually, we do not optimize directly the AP since gradient boosting requires the use of differentiable loss functions as metionned in the related part of Section 1.3. We rather introduce a differentiable approximation of AP that can then be optimized in a gradient boosting algorithm.

3.3 Stochastic gradient boosting with AP

In this section, we recall the stochastic gradient boosting framework as presented by (Friedman, 2002) and already introduced in Section 1.3 of this document. Then we instantiate the loss function in two different ways: first, we introduce a differentiable version of AP using the sigmoid function. Then, in order to reduce the algorithmic complexity, we suggest using a rough approximation based on the exponential function. We show that this second strategy allows us not only to drastically reduce the complexity but also, to get similar or even better results than the sigmoid-based loss. We give some explanations about this behaviour at the end of the section.

3.3.1 Stochastic gradient boosting

Like other boosting methods, gradient boosting is based on a sequential and adaptive learning over weak learners that are linearly combined. However, instead of setting a weight for every example, gradient boosting builds each new weak learner on the residuals of the previous linear combination. We can see gradient boosting as gradient descent in functional space. The linear combination at step t is defined as follows:

$$F_t(x) = F_{t-1}(x) + \alpha_t h_t(x),$$

with $h_t \in \mathcal{H}$ a hypothesis belonging to a class of models $\mathcal{H}$ (typically, regression trees) and α_t the weight underlying the performance of h_t in the linear combination. Residuals are defined by the negative gradients of the loss function computed w.r.t. the previous linear combination of weak learners:

$$r_t^i = -\Big[\frac{\partial \ell(z_i, f_{t-1}(x_i))}{\partial f_{t-1}(x_i)}\Big], i = 1 \dots M.$$

As in standard boosting, hard examples get more importance along the iterations of gradient boosting. Note that a mini-batch strategy is usually used to speed-up the procedure by randomly selecting a proportion $\lambda \in [0, 1]$ of examples at each iteration. Additionally, this stochastic approach allows us to avoid falling in a local optima. A generic version of the stochastic gradient boosting is presented in Algorithm 5.

Algorithm 5 Stochastic gradient boosting

INPUT: a training set $S = \{z_i = (x_i, y_i)\}_{i=1}^{M}$, a parameter $\lambda \in [0, 1]$, a weak learner

Require: Initialize $F_0(x) = 0$

for $t = 1$ to T **do**

Select randomly from S a set $S' = \{x_i, y_i\}_{i=1}^{\lambda M}$

$$r_t^i = -\Big[\frac{\partial \ell(z_i, F_{t-1}(x))}{\partial F_{t-1}(x_i)}\Big], \forall z_i = (x_i, y_i) \in S' \tag{3.8}$$

Fit a weak classifier (*e.g.* a regression tree) $h_t(x)$ to predict the targets r_t

Find $\alpha_t = argmin_\alpha \sum_{i=1}^{M} \ell(z_i, F_{t-1}(x_i) + \alpha h_t(x_i))$

Update $F_t(x)$ such that $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$

end for

Output the final model:

$$F^*(x) = sign(\sum_{t}^{T} f_t(x)).$$

The key step of this algorithm takes place in Eq. (3.8). It requires the definition of a differentiable loss function with its associated gradients. Unlike the state of the art ranking methods which make use of gradient boosting, we aim at directly optimizing in the loss function ℓ a surrogate of AP.

3.3.2 Sigmoid-based Surrogate of AP

To define a loss function ℓ based on AP, we need to transform the non-differentiable Eq.(3.6) into an expression for which one will be able to compute gradients on AP. Therefore, we need to get rid of the indicator function. A standard way consists in replacing $\mathrm{I}(f(x_i) \leq f(x_j))$ by the sigmoid function :

$$\mathrm{I}(f(x_i) \leq f(x_j)) \approx \frac{1}{1 + e^{-\alpha(f(x_j) - f(x_i))}} = \sigma(f(x_j) - f(x_i))$$

with α a smoothing parameter. As α grows the approximation gets closer to the true AP. Considering that $\sum_{j=1}^{M} \mathrm{I}(y_j = 1) = P$, we get the following differentiable surrogate of AP:

$$\hat{AP}_{sig} = \frac{1}{P} \sum_{i=1}^{P} \frac{1}{\sum_{j=1}^{M} \frac{1}{1 + e^{-\alpha(f(x_j) - f(x_i^+))}}} \sum_{j=1}^{P} \frac{1}{1 + e^{-\alpha(f(x_j^+) - f(x_i^+))}}$$

$$
\begin{aligned}
&= \frac{1}{P}\sum_{i=1}^{P}\frac{\sum_{j=1}^{P}\sigma(f(x_j^+)-f(x_i^+))}{\sum_{h=1}^{M}\sigma(f(x_h)-f(x_i^+))} \\
&= \frac{1}{P}\sum_{i=1}^{P}\hat{p}(k_i) \approx \frac{1}{P}\sum_{i=1}^{P}p(k_i). \qquad (3.9)
\end{aligned}
$$

From $\hat{AP}_{sig}$, we get the following objective function:

$$1-\hat{AP}_{sig} = \mathbb{E}_{z_i\in S^+}\ell_{ap}^{sig}(z_i,f),$$

where $\ell_{ap}^{sig}(z_i,f) = 1-\hat{p}(k_i)$ is the loss suffered by f in terms of precision at z_i (let us remind that k_i is the rank (predicted by f) of the i^{th} positive example z_i). In fact, we can simply rewrite our objective function as:

$$1-\hat{AP}_{sig} = \frac{1}{P}\sum_{i=1}^{P}\frac{\sum_{j=1}^{N}\sigma(f(x_j^-)-f(x_i^+))}{\sum_{h=1}^{M}\sigma(f(x_h)-f(x_i^+))}$$

For the sake of simplicity, let us use the following notations:
$\sigma(f(x_j)-f(x_i)) = \sigma_{ji}$ and we have

$$
\begin{aligned}
\frac{\partial\sigma_{ji}}{\partial f_t(x_j)} &= -\sigma_{ji}(1-\sigma_{ji}) = -\sigma'_{ji}, \\
\frac{\partial\sigma_{ji}}{\partial f_t(x_i)} &= \sigma_{ji}(1-\sigma_{ji}) = \sigma'_{ji}.
\end{aligned}
$$

The gradient w.r.t $f_t(x_p^+)$ or $f_t(x_p^-)$, for positive and negative examples respectively are given by:

$$
\begin{aligned}
\frac{\partial(1-\hat{AP}_{sig})}{\partial f_t(x_p^+)} &= \frac{\partial(1-\hat{AP}_{sig})}{\partial\sigma_{jp}}\frac{\partial\sigma_{jp}}{\partial f_t(x_p^+)} + \frac{\partial(1-\hat{AP}_{sig})}{\partial\sigma_{pi}}\frac{\partial\sigma_{pi}}{\partial f_t(x_p^+)} \\
&= \sum_{j=1}^{P}\frac{(\sigma'_{jp}\sum_{h=1}^{M}\sigma_{hp} - \sigma_{jp}\sum_{h=1}^{N}\sigma'_{hp})}{(\sum_{h=1}^{N}\sigma_{hp})^2} \\
&\quad + \sum_{i=1}^{P}\frac{(\sigma'_{pi}\sum_{h=1}^{N}\sigma_{hi} - \sigma_{jp}\sigma'_{pi})}{(\sum_{h=1}^{N}\sigma_{hi})^2}, \qquad (3.10)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial(1-\hat{AP}_{sig})}{\partial f_t(x_p^-)} &= \frac{\partial(1-\hat{AP}_{sig})}{\partial\sigma_{pi}}\frac{\partial\sigma_{pi}}{\partial f_t(x_p^-)} \\
&= \sum_{i=1}^{P}\sum_{j=1}^{P}\frac{-\sigma_{ji}\sigma'_{pi}}{(\sum_{h=1}^{N}\sigma_{hi})^2},
\end{aligned}
$$

as, $\frac{\partial(1-\hat{AP}_{sig})}{\partial\sigma_{jp}}\frac{\partial\sigma_{jp}}{\partial f_t(x_p^-)} = 0$, since the example x_p from the previous formulation will always be positive in $1-\hat{AP}$.

The proposed approximation of AP presented above makes use of a sigmoid-based approximation of the indicator function. We denote as SGBAP$_{sig}$, the stochastic Gradient Boosting algorithm using this sigmoid-based approximation. While this approach allows us to have a differentiable approximation of AP, it has the drawback to require a quadratic time for being computed which can be intractable on very large scale datasets. In the next subsection, we discuss another approximation of AP able to be computed more efficiently.

3.3.3 Exponential-based Surrogate of AP

A quadratic computation time for the estimation we presented in the previous section may be a too strong algorithmic constraint to deal with real-world applications (*e.g.* fraud detection in credit card transactions contains millions of data points). To overcome this issue, we suggest here to resort to a less costly surrogate of AP using the exponential function as an approximation of the indicator function.

$$\mathrm{I}(f(x_i) \leq f(x_j)) \approx e^{(f(x_j)-f(x_i))}.$$

As already done in Rankboost (Freund et al., 2003), we can show that the use of this exponential function allows us to reduce the time complexity for binary datasets to $\mathcal{O}(P+N)$.

Using the new approximation, AP takes the following form:

$$\begin{aligned}\hat{AP}_{exp} =& \frac{1}{P}\sum_{i=1}^{P}\frac{\sum_{j=1}^{P} e^{f(x_j^+)}e^{-f(x_i^+)}}{\sum_{h=1}^{M} e^{f(x_h)}e^{-f(x_i^+)}} = \frac{1}{P}\sum_{i=1}^{P}\frac{e^{-f(x_i^+)}\sum_{j=1}^{P} e^{f(x_j^+)}}{e^{-f(x_i^+)}\sum_{h=1}^{M} e^{f(x_h)}} \\ =& \frac{\sum_{j=1}^{P} e^{f(x_j^+)}}{\sum_{h=1}^{M} e^{f(x_h)}}\end{aligned}$$

as for the sigmoid approximation, we rather use $1-\hat{AP}_{exp}$ to minimize it.

$$1-\hat{AP}_{exp} = \frac{\sum_{h=1}^{M} e^{f(x_h)} - \sum_{j=1}^{P} e^{f(x_j^+)}}{\sum_{h=1}^{N} e^{f(x_h)}} = \frac{\sum_{n=1}^{N} e^{f(x_n^-)}}{\sum_{h=1}^{M} e^{f(x_h)}} \tag{3.11}$$

Finally, finding the gradients of this new objective function is straightforward.

$$\begin{aligned}\frac{\partial 1-\hat{AP}_{exp}}{\partial f(x_p^+)} &= \frac{-e^{f(x_p^+)}\sum_{n=1}^{N} e^{f(x_n^-)}}{(\sum_{h=1}^{M} e^{f(x_h)})^2} \\ \frac{\partial 1-\hat{AP}_{exp}}{\partial f(x_p^-)} &= \frac{e^{f(x_p^-)}\sum_{i=1}^{M} e^{f(x_h)} - e^{f(x_p^-)}\sum_{n=1}^{N} e^{f(x_n^-)}}{(\sum_{h=1}^{M} e^{f(x_h)})^2}\end{aligned} \tag{3.12}$$

In the following, we call our method SGBAP, the stochastic gradient boosting based on our approximation $1-\hat{AP}_{exp}$.

Note that in Eq. 3.12, one can see an adverse effect brought by the exponential approximation of the indicator function. Indeed, if $f(x_i)$ is first in the ranking, the gradient of x_i, $g(x_i)$, should decrease as there is no other position in which it will improve the overall AP. However, in our approximation, when $f(x_i)$ is significantly high, the gradient for this example will be the highest.

Assume $\forall j \in \mathcal{S} \setminus x_i, f(x_i) >> f(x_j)$, we have $g(x_i) \approx 1$ and $g(x_k) \approx 0 \quad \forall k \in \mathcal{S}^- \setminus x_i$. In fact, this effect is limited with stochastic gradient boosting. Indeed, since $g(x_i)$ is not computed during all the iteration thanks to the random mini-batches, the gradient is then automatically regularized. However, running the gradient boosting algorithm instead of the stochastic version would raise the previous effect. The same holds for any basic gradient descent based algorithm.

3.3.4 Comparison between the Approximations of AP

In this section, we compare experimentally the approximations used in this paper - $\hat{AP}_{exp}$ and $\hat{AP}_{sig}$ - with a simple one-dimensional sample described in Table 4.1. For this experiment, we use a simple linear model $f(x) = \theta_0 + \theta_1 x$. The toy dataset has been made such that the model has three ranking choices: (i) rank the examples in descending order from $x = +7$ to $x = -6$ (when $\theta_1 > 0$), (ii) rank the examples in descending order from $x = -6$ to $x = +7$ ($\theta_1 < 0$) or (iii) give the same rank to every example ($\theta_1 = 0$). We give the AP and $AUCROC$ measures in each case : $AP = 0.29$, $AUCROC = 0.52$ when $\theta_1 < 0$, $AP = 0.33$, $AUCROC = 0.49$ when $\theta_1 > 0$ and $AP = 0.22$, $AUCROC = 0.5$ when $\theta_1 = 0$. Figure 3.3, shows that the two objective functions considered are obviously not convex. However, they both find their minimum in $\theta_1 > 0$ which yields the best AP.

Note that on Figure 3.3, $1-\hat{AP}_{exp}$ has another advantage than the time complexity over the sigmoid approximation. Indeed, for negative examples with high scores

Table 3.1: Toy dataset constituted of 14 examples on the real line with their associated labels. x correspond to the feature value and y the class.

x	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7
y	−1	−1	−1	+1	+1	−1	−1	−1	−1	−1	−1	−1	+1	−1

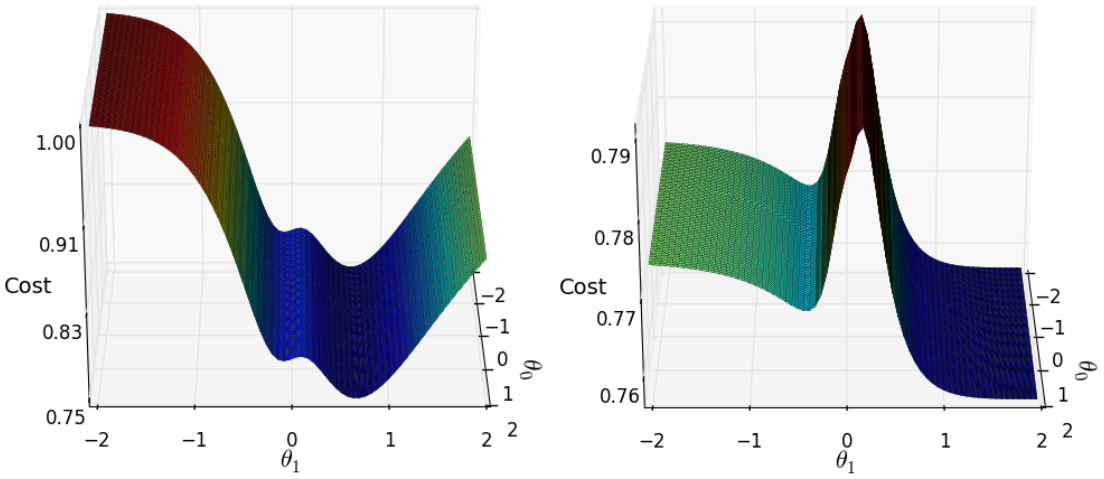

Figure 3.3: $1 - \hat{AP}_{exp}$ (on the left) and $1 - \hat{AP}_{sig}$ (on the right) costs in function of the two model parameters θ_0 and θ_1.(better with colour)

(*e.g.* when $\theta_1 > 1$), $1-\hat{AP}_{sig}$ tends to have vanishing gradients while, for $1-\hat{AP}_{exp}$, they tend to increase exponentially. Indeed, on Figure 3.3, the cost increases for the exponential approximation while it decreases for the sigmoid approximation.

Figure 3.4 presents the pairwise cost function based on $AUCROC$ and a surrogate of the accuracy, the logistic loss. The minimum for these two functions is reached for a $\theta_1 < 0$ which reverses the ranking compared to the one obtained by optimizing a surrogate of AP. Interestingly, all ranking based surrogate functions do not depend on θ_0. Only the logistic loss make use of this parameter to correct the classification regarding the decision threshold.

3.4 Experiments

In this section, we present an experimental evaluation of our approach in two parts. In a first setup, we provide a comparative study with different state-of-the-art methods and various evaluation measures on 5 imbalanced public datasets coming either from the UCI Irvine Machine Learning repository or the LIBSVM datasets [1] and on a real dataset of credit card transactions provided by the private

[1] `http://archive.ics.uc/du/ml/` and `https:/www.csie.ntu.edu.tw/cjlin/libsvmtools/`

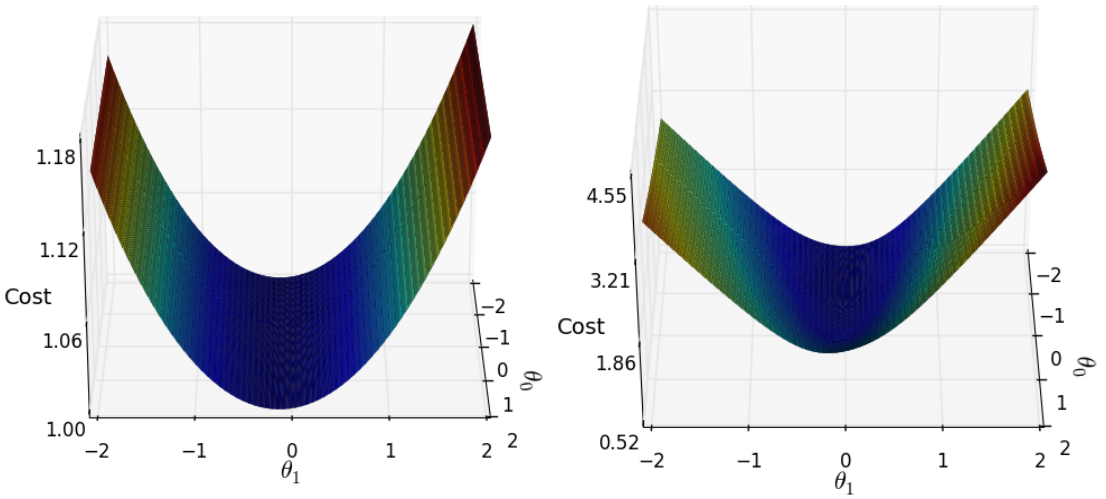

Figure 3.4: Rankboost objective function (on the left) and the logistic loss based objective function (on the right). They find their minimum in $\theta_1 = -0.14$ and $\theta_0 = -1.28 \quad \theta_1 = -0.02$ respectively.

company Worldline that is representative of the industrial application introduced in Chapter 2. In a second experiment, we study the robustness of our method to undersampling of positive instances.

3.4.1 Top-rank quality over imbalanced datasets

In this experiment, we use the public datasets Pima, Breast cancer, HIV, Heart Cleveland, w8a and the real fraud detection dataset over credit card transactions provided by Worldline. This dataset contains 2 million transactions labelled as 1 fraudulent or -1 genuine where 0.2% are fraudulent. It is constituted of 2 subsets of transactions of 3 consecutive days each. The first one is fixed as the training set and the second as the test. Each subset being separated by one week in order to have the same week days (*e.g.* Tuesday to Thursday) in train and test. This setting models a realistic scenario where the feedback for every transaction is obtained only few days after the transaction was performed. The properties of the different datasets are summarized in Table 4.4.
We now describe our experimental setup. For the public datasets where the training/testing sets are not available directly, we randomly generate 2/3-1/3 splits of the data to obtain training and test sets respectively. Hyperparameters are tuned thanks to a 5-fold cross-validation over the training set, keeping the values offering the best AP. We repeat the process over 30 runs and average the results.

We compare our method, named SGBAP, to 4 other baselines[2]: SGBAP$_{sig}$ as de-

[2]Note that we did not use AdaRank in our evaluation because the weight updates rely on a

Table 3.2: Properties of the 6 datasets used in the experiments.

	#examples	Positives ratio	#Features
Pima	767	34%	8
Breast cancer	286	30%	9
HIV	3,272	13.3%	8
Heart cleveland (4 vs all)	303	4.3%	13
w8a	64000	3%	300
Fraud	2,000,000	0.2%	40

fined previously, GB-Logistic which is the basic gradient boosting with a negative binomial log-likelihood loss function (Friedman, 2001) (pointwise and accuracy based for binary datasets), LambdaMART-AP (Wu et al., 2010) a version of gradient boosting that optimizes the average precision and RankBoost (Freund et al., 2003), a pairwise version of AdaBoost for ranking. For each method, we fix a time limit to 86,000sec.

We evaluate the previous methods according to 4 criteria measured on the test sets. First, we use the classic average precision (AP) and $AUCROC$. Additionally, we also consider 2 measures to better assess the quality of the approaches for top-rank precision. For this purpose, we use the performance $Pos@Top$, defined in (Li et al., 2014a), that gives the percentage of positive example retrieved before a negative appears in the ranking. In other words, it corresponds to the recall before the precision drops under 100%.

We also evaluate the $P@k$ from Eq. 3.4. In our setup, we set k being the number of positive examples, which makes sense in our context of highly imbalanced data when the objective is to provide a short list of alerts to an expert and where the number of positive examples is much smaller than the negative examples. In fact, the latter measure is both precision and recall at rank k.

The results obtained are reported on Table 3.3. First, we can remark that except for the Pima dataset that has the highest positive ratio, our approach is always better in terms of AP. SGBAP is also better than other baselines in terms of $Pos@top$ which is the hardest measure for evaluating the top-rank performance. Additionally, we see that for all datasets with a significantly low positive ratio (less than 15%), our approach is always better according to the P@k measure. Overall, we can remark that when the imbalance is high, our approach is always significantly better than other baselines according to 3 criteria: AP, $Pos@top$ and

notion of query that is not adapted to our framework.

$P@k$ which clearly confirms that our method performs better for optimizing top-rank results. Note that, for the dataset HIV, SGBAP$_{sig}$ performed quite poorly. We believe that this is because of the early vanishing gradient due to the imbalance in the dataset. This effect does not appear in heart cleveland dataset most likely because of the small dataset size.

Table 3.3: Results obtained for the different evaluation criteria used in the paper. We indicate in bold font the best method with respect to each dataset and each evaluation measure. A $-$ indicates that the method did not finish before the time limit.

Dataset	Algorithm	*AUCROC*	*AP*	*Pos@Top*	*P@k*
Pima	GB-Logistic	0.8279 ± 0.0352	0.7125 ± 0.0267	0.0388 ± 0.0379	$\mathbf{0.6608 \pm 0.0296}$
	RankBoost	$\mathbf{0.8352 \pm 0.0359}$	0.7281 ± 0.0621	$\mathbf{0.0620 \pm 0.0546}$	0.6586 ± 0.0298
	LambdaMART-AP	0.8177 ± 0.0304	$\mathbf{0.7338 \pm 0.0528}$	0.0407 ± 0.0443	0.6559 ± 0.0257
	SGBAP	0.8276 ± 0.0418	0.7119 ± 0.0486	0.0579 ± 0.0577	0.6455 ± 0.0356
	SGBAP$_{sig}$	0.8215 ± 0.0215	$0.7091 \pm, 0.0328$	0.0388 ± 0.0346	0.6514 ± 0.0325
Breast cancer	GB-Logistic	0.6821 ± 0.0756	0.5089 ± 0.0562	0.0931 ± 0.0561	0.4457 ± 0.0739
	RankBoost	0.6492 ± 0.0562	0.4838 ± 0.0632	0.0461 ± 0.0513	0.4626 ± 0.0629
	LambdaMART-AP	0.6733 ± 0.0419	0.5280 ± 0.0680	0.0859 ± 0.0828	$\mathbf{0.5196 \pm 0.0624}$
	SGBAP	0.7124 ± 0.0596	$\mathbf{0.5602 \pm 0.0830}$	$\mathbf{0.1019 \pm 0.1018}$	0.4980 ± 0.0612
	SGBAP$_{sig}$	$\mathbf{0.7131 \pm 0.0521}$	0.5503 ± 0.0443	0.0729 ± 0.0693	0.5061 ± 0.0574
HIV	GB-Logistic	0.8598 ± 0.0155	0.5557 ± 0.0376	0.0303 ± 0.0284	0.5391 ± 0.0364
	RankBoost	0.8599 ± 0.0127	0.5464 ± 0.0276	0.0401 ± 0.0363	0.5309 ± 0.0254
	LambdaMART-AP	0.8222 ± 0.0466	0.4286 ± 0.0887	0.0075 ± 0.0176	0.4874 ± 0.0814
	SGBAP	$\mathbf{0.8661 \pm 0.0150}$	$\mathbf{0.5737 \pm 0.0347}$	$\mathbf{0.0536 \pm 0.0410}$	$\mathbf{0.5445 \pm 0.0351}$
	SGBAP$_{sig}$	0.7578 ± 0.0231	0.3928 ± 0.0434	0.041 ± 0.0250	0.3902 ± 0.0439
Heart cleveland	GB-Logistic	0.7544 ± 0.1020	0.1638 ± 0.0931	0.0133 ± 0.0498	0.1 ± 0.1420
	Rankboost	$\mathbf{0.8109 \pm 0.0515}$	0.1739 ± 0.0638	0.0150 ± 0.0565	0.0967 ± 0.1335
	LambdaMART-AP	0.7277 ± 0.1225	0.1809 ± 0.1011	0.0383 ± 0.0863	0.1333 ± 0.1287
	SGBAP	0.7789 ± 0.1178	$\mathbf{0.2188 \pm 0.1103}$	$\mathbf{0.0483 \pm 0.0970}$	$\mathbf{0.2017 \pm 0.1044}$
	SGBAP$_{sig}$	0.7983 ± 0.0638	0.2136 ± 0.0964	0.045 ± 0.0906	0.1566 ± 0.1295
w8a	GB-Logistic	0.9544 ± 0.0039	0.7385 ± 0.0154	0.0534 ± 0.0529	0.7091 ± 0.0152
	RankBoost	$\mathbf{0.9712 \pm 0.0028}$	0.7649 ± 0.0135	0.0392 ± 0.0451	0.7277 ± 0.008
	LambdaMART-AP	$-$	$-$	$-$	$-$
	SGBAP	0.9701 ± 0.0029	$\mathbf{0.8351 \pm 0.0100}$	$\mathbf{0.1779 \pm 0.0978}$	$\mathbf{0.7972 \pm 0.0132}$
	SGBAP$_{sig}$	$-$	$-$	$-$	$-$
Fraud	GB-Logistic	0.8808	0.1477	0.0009	0.2411
	RankBoost	**0.8829**	0.1560	0.0005	0.2449
	LambdaMART-AP	$-$	$-$	$-$	$-$
	SGBAP	0.6878	**0.1747**	**0.0059**	**0.3203**
	SGBAP$_{sig}$	$-$	$-$	$-$	$-$

3.4.2 Top rank capability for a decreasing positive ratio

In this section, we present an experiment showing the robustness of our approach when the number of positive examples decreases. We consider the Pima dataset

because it has the highest ratio of positive instances and because our approach did not perform the best for all criteria. We aim at under-sampling the positive class (*i.e.* to decrease the positive ratio $\frac{P}{M}$). We start from the original positive ratio (34%) and go down to 3% by steps of ~ 0.05. For every new dataset, we follow the same experimental setup as described previously. At the end of the 30 runs for a given positive ratio dataset, we compute the average rank obtained by the examples in the test set and remove the top k positive instances such that $\frac{P-k}{M}$ is equal to the next positive ratio to evaluate. We repeat the previous set up until we reach 3% of positive examples in the dataset. We repeat this process independently for each method. The objective is to remove from the current dataset the easiest positive examples for each approach to evaluate its capability to move to the top new positive examples. Note that this makes harder the problem of ranking correctly in the top positive instances. Thus, the top rank performance measures should globally decrease.

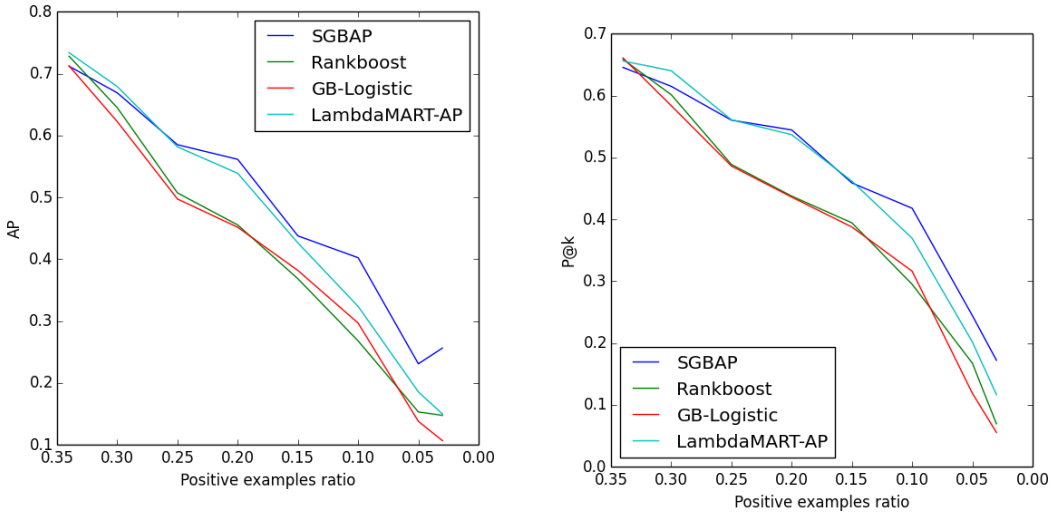

Figure 3.5: The average precision and P@k at different positive example ratio for pima dataset.

The results with respect to the AP criterion and $P@k$ are presented on Figure 3.5. From this experiment, we see that SGBAP outperforms the other models as the imbalance ratio increases and notably when the ratio of positives becomes smaller than 15% which confirms that our approach behaves clearly the best when the level of the imbalance is high in comparison to other state of the art approaches.

3.5 Conclusion and Perspectives

In this paper, we presented SGBAP, a novel Stochastic Gradient Boosting based approach for optimizing directly a surrogate of the average precision measure. Our approximation is based on an exponential surrogate allowing us to compute our criterion in linear time which is crucial for dealing with large scale datasets such as for fraud detection tasks. We claim that this approach is well adapted for supervised anomaly detection in the context of highly imbalanced settings. Indeed, our criterion focuses specifically on the top-rank yielding a better *precision* in the top k positions.

A perspective of this work would be to optimize other interesting measures for learning to rank such as NDCG by means of a stochastic gradient descent approach. Another direction would be to adapt the optimization of the surrogate of average precision to other learning models such as neural networks where we could take benefit from recent results in non-convex optimization. There is also interesting modifications of the AP Flach and Kull (2015) that benefit from different advantages (mainly, the interpolation between points and the invariance of the metric for different class distribution) and that could form an interesting loss to use as an objective function.

In this work we observed that specific weak learners or a combination of weak learner could achieve higher precision on the precision and recall curve. Due to the linear combinations in gradient boosting, this performance is averaged with all the other learners. This prompted us to in a different direction, where we use gradient boosting to learn non-linear combinations of weak learners instead of the linear combination usually considered. This is the objective of the contribution presented in the next chapter of this thesis.

Chapter 4

Non-Linear Gradient Boosting in Multi-Latent Spaces

This chapter is based on the following publication

Frery, Jordan, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. "Online Non-Linear Gradient Boosting in Multi-Latent Spaces" In Intelligent Data Analysis. 2018

Abstract

Gradient Boosting is a popular ensemble method that combines linearly diverse and weak hypotheses to build a strong classifier. In this work, we propose a new Online Non-Linear gradient Boosting (ONLB) algorithm and its batch counterpart, NLB, where we suggest to jointly learn different combinations of the same set of weak classifiers in order to learn the idiosyncrasies of the target concept. To expand the expressiveness of the final model, our method leverages the non-linear complementarity of these combinations. We perform an experimental study showing that ONLB (i) outperforms most recent online boosting methods in terms of both convergence rate and accuracy and (ii) learns diverse and useful new latent spaces. Moreover, we present an experimental study for class imbalance problems and show that NLB outperforms the linear version of gradient boosting.

4.1 Introduction

Apart from the class imbalance problem, real life applications such as fraud detection, click prediction, spam detection and face recognition have a specificity rather

overlooked: the uninterrupted flow of data. As machine learning gains popularity in the industry, one must consider the problem of training models over always increasing volumes of data that always need more memory and more storage. While big data centres can partially solve the memory problem, training the model from scratch each time new instances arrive remains unrealistic.

To overcome this problem, online boosting has received much attention during the past few years (Oza, 2005; Grabner and Bischof, 2006; Chen et al., 2012; Beygelzimer et al., 2015b; Jung et al., 2017; Beygelzimer et al., 2015a; Hu et al., 2017). In these methods, the boosted model is updated after seeing one example. While they can process efficiently large amount of data, their practical limitations include: (i) an edge assumption usually made on the asymptotic accuracy (*i.e.* the edge over random guessing) of the weak learners making some approaches hard to tune (ii) the absence of a weighting scheme of the weak learners that depends on their performance and (iii) for some of them, a lack of adaptiveness (despite the fact that it was a strong point of Adaboost (Freund and Schapire, 1997)).

Moreover, all the previous online methods face another issue: they usually perform a linear combination over a finite number of learned hypotheses which may limit the expressiveness of the final model to reach complex target concepts.

In the previous chapter, we used a linear gradient boosting (GB) to optimize the average precision with a new objective function. While working with GB, it sometimes appears that combining linearly the weak learners outputs was not optimal. In Figure 4.1 we show this phenomenon on a two-dimensional toy dataset. The right figure shows the optimal probability boundaries of linear GB with two decision stumps. We clearly see that a huge mistake is made on the upper left corner by assigning high probability to these examples to belong the red class while most of them are from the blue class. We will come back to this example in Section 4.5.

Another limitation of GB is its adaptation to the online setting. While the batch setting would allow us to add step by step new hypotheses and capture the complexity of the underlying problem, an online algorithm keeps the same set of weak learners all along the process. This remark prompted us to investigate the way to develop a **non-linear** gradient boosting algorithm with an enhanced expressiveness. To the best of our knowledge, there is only one work specific to non-linear boosting (García-Pedrajas et al., 2007) but only usable in a batch setting. This is why the main contribution of this chapter takes the form of a new algorithm, called ONLB - for Online Non-linear gradient Boosting.

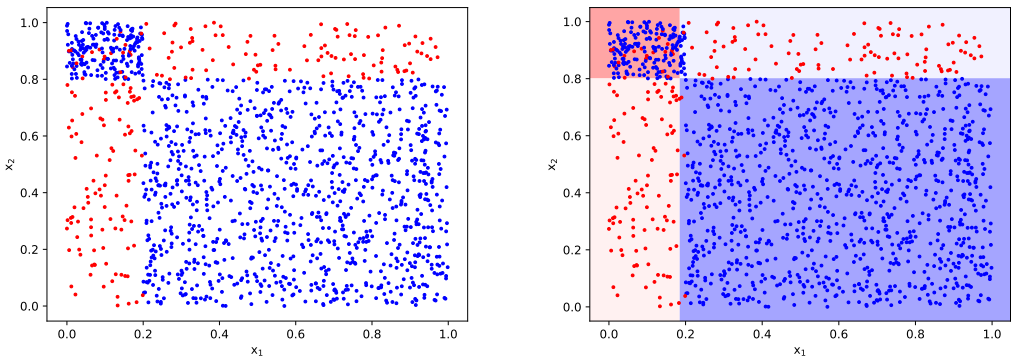

Figure 4.1: On the left, we present the toy dataset with two classes. The red class is in minority. The figure on the right shows the probability boundaries of GB on the test set. Blue areas show a strong probability for the examples to belong to the blue class while the red areas show a strong probability for the examples to belong to the red class. As the colour disappears, the model is uncertain to which class the examples belong. The model used here is a linear gradient boosting with two decision stumps.

Inspired from previous research in domain adaptation (Becker et al., 2013), boosted-multitask learning (Chapelle et al., 2011) and boosting in concept drift (Scholz and Klinkenberg, 2007), ONLB resorts to the same set of boosted weak learners, projects their outputs in different latent spaces and takes advantage of their complementarity to learn non-linearly the idiosyncrasies of the underlying concept. ONLB is illustrated in Figure 4.2. At first glance, it looks similar to boosted neural networks, as done in (Han et al., 2016; Opitz et al., 2017), where the embedding layer is learned with boosting in order to infer more diversity. However, our method aims at learning the weak hypotheses iteratively where the following weak learner tries to minimize the error made by the network restricted to the previous hypotheses (see the solid lines in Figure 4.2).

The other main difference comes from the back-propagation that is performed at each step only on the parameters related to the weak learner subject to an update (see the red lines in Figure 4.2). Thanks to the non-linear function brought by the last layer to combine the different representation output, ONLB converges much faster than the other state of the art online boosting algorithms.

The chapter is organized as follows: Section 4.2 is devoted to the presentation of the related work. Our new non-linear online gradient boosting algorithm ONLB is presented in Section 4.3 and its batch counterpart in Section 4.4. Section 4.5 is dedicated to a large experimental comparison with the state of the art methods

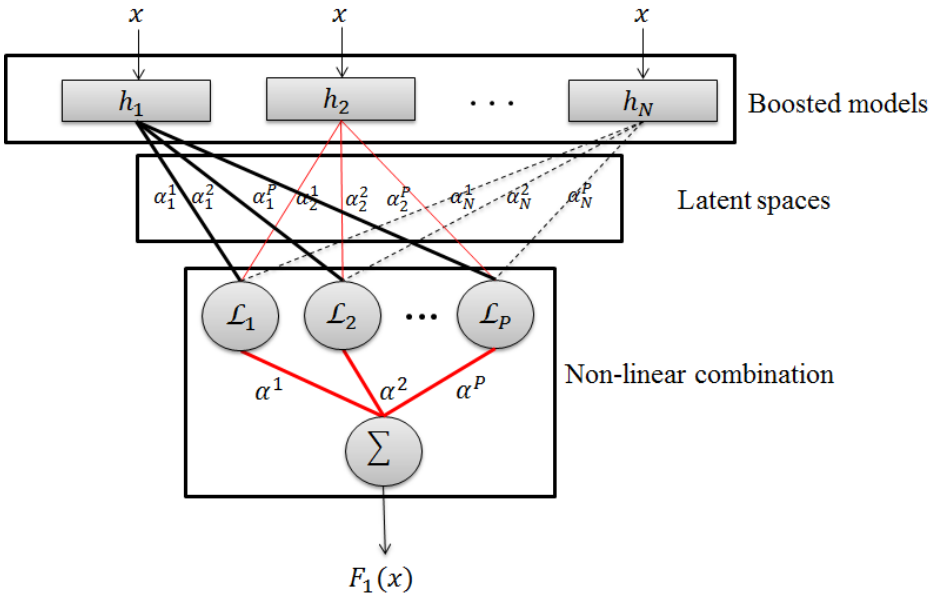

Figure 4.2: Graphical representation of our Online Non-Linear gradient Boosting method: the first top layer corresponds to the learned weak classifiers; the second layer represents different linear combinations of their outputs; the bottom layer proceeds a non-linear transformation of those combinations. The thickest lines show the needed activated path to learn a given classifier (here h_2). The red lines show the update performed only on the parameters concerned by this weak learner. The dashed lines are not taken into account at this iteration.

where we also provide an evidence for NLB to outperform linear gradient boosting on imbalanced datasets. We conclude the chapter in Section 4.6.

4.2 Related work

Online boosting methods have been developed soon after their batch counterpart. The first one introduced in (Oza, 2005) uses a resampling method based on a Poisson distribution and was applied in computer vision by (Grabner and Bischof, 2006) for feature selection. Theoretical justifications were developed later in (Chen et al., 2012) where they notably discuss the number of weak learners needed in an online boosting framework. This is indeed a major concern since having too many of them could lead to predictions dominated by redundant weak learners that perform poorly. On the other hand, too few weak learners could make the boosting process itself irrelevant, as the goal is still to improve upon the performance of a simple base learner. More recently, (Beygelzimer et al., 2015b) extends this previous work to propose an optimal version of boosting in terms of the number

of weak learners for classification.

An adaptation of this framework to multiclass online boosting was proposed in (Jung et al., 2017). While these methods come with a solid theory, the assumption usually made on the asymptotic accuracy of the weak learners leads to two main practical limitations. The first one is the undeniable difficulty to estimate this edge without prior knowledge on the task at hand. The second comes from the fact that the edge of each weak learner might be very different depending on their own performance. And it turns out that the latter is never taken into consideration and might impact the overall performance of boosting.

Online gradient boosting was introduced by (Leistner et al., 2009) allowing one to use more general loss functions but without any theoretical guarantees. Later, Beygelzimer et al. (2015a) and its extension to non-smooth losses (Hu et al., 2017), propose online gradient boosting algorithms with theoretical justifications. However, the linear aspect of these methods limit their expressiveness strongly.

Another series of related works is the use of boosting in neural network methods. Recently, neural networks were used with incremental boosting (Han et al., 2016) to train a specific layer. In Opitz et al. (2017), the authors built upon Beygelzimer et al. (2015a) to optimize and increase the diversity of their embedding layer. Our work is related in the sense that we boost a layer to build a new feature space. However, the main goal is not to learn a general neural network. This layer is rather used to make connections between our different weak learners. This is why our back-propagation procedure differs by focusing only on the parameters of the weak learner to be optimized at each step.

Apart from online boosting methods, our work is also related to non-linear boosting. However, as far as we know, only (García-Pedrajas et al., 2007) tackled this topic by proposing a non-linear boosting projection method where, at each iteration of boosting, they build a new neural network only with the examples misclassified at the previous round. They finally take the new feature space induced by the hidden layer and feed it as the input space for the next learner. Nonetheless, it is very expensive and unsuitable to online learning.

4.3 Online Non-Linear gradient Boosting

In this study, we consider a binary supervised online learning setting where at each time step $i = 1, 2, ..., M$ one receives a labelled example $(x_i, y_i) \in \mathcal{X} \times \{-1, 1\}$ where $\mathcal{X}$ is a feature space. In this setting, the learner makes a prediction $f(x_i)$, the true label y_i is then revealed and it suffers a loss $\ell(f(x_i), y_i)$.

Boosting aims at combining different weak hypotheses. In batch gradient boosting, weak learners are learned sequentially while in the online setting, they are not allowed to see all examples at once. Thus, it is not possible to simply add new models iteratively in the combination as in the batch boosting. In fact, online boosting maintains a sequence of T weak online learning algorithms $\mathcal{A}_1, ..., \mathcal{A}_T$ such that each weak learner h_t is updated by $\mathcal{A}_t$ in an online fashion. Note that every $\mathcal{A}_t$ considers hypotheses from a given restricted hypothesis class $\mathcal{H}$. The final model corresponds to a weighted linear combination of the T weak learners:

$$F(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \tag{4.1}$$

where α_t stands for the weight of the weak learner h_t.

We now present our Online Non-Linear gradient Boosting, ONLB. As shown in Figure 4.2, our method maintains $\mathcal{P}$ different representations that correspond to different combinations of the T learned weak learners, projecting their outputs into different latent spaces. Every representation p is updated right after a weak learner is learned. The outputs given by the p representations are then merged together to build a strong classifier, $F(x)$. To capture non-linearity during this process, we propose to pass the output of each representation p into a non-linear function $\mathcal{L}_p$. We define the prediction of our model $F(x)$ as follows:

$$F(x) = \sum_{p=1}^{\mathcal{P}} \alpha^p \mathcal{L}_p\Big(\sum_{t=1}^{T} \alpha_t^p h_t(x) \Big), \tag{4.2}$$

where α_t^p is the weight projecting the outputs of the weak learner h_t in the latent space p and α^p the weight of this representation. Eq (4.2) illustrates clearly the difference with linear boosting formulation of Eq (4.1). We denote by F_k the classifier restricted to the first k weak learners: $F_k(x) = \sum_{p=1}^{\mathcal{P}} \alpha^p \mathcal{L}_p\Big(\sum_{t=1}^{k} \eta \alpha_t^p h_t(x) \Big)$.

Our method aims thus at combining the same set of classifiers into different latent spaces. A key point here relies in making these classifiers diverse while still being relevant in the final decision. To achieve this goal, we update every weak learner h_t to decrease the error of the previous model F_{t-1} such that:

$$h_t = argmin_h \sum_{i=1}^{M} \ell_c\big(\sum_{p=1}^{\mathcal{P}} \alpha^p \mathcal{L}_p\big(\sum_{k=1}^{t-1} \alpha_k^p h_k(x_i) + h(x_i) \big), y_i \big), \tag{4.3}$$

where $\ell_c(\cdot, \cdot)$ is a classification loss. In other words, we look for a learner h_t able to improve the current model F_{t-1}.

In gradient boosting (Friedman, 2001), one way to learn the following weak learner is to approximate the negative gradient (residuals) of F_{t-1} by minimizing the square loss between these residuals and the weak learner predictions. We define r_t^i the residual at iteration i for the example x_i as follows:

$$r_t^i = -\frac{\partial \ell_c(F_{t-1}(x_i), y_i))}{\partial F_{t-1}(x_i)}. \tag{4.4}$$

In fact, from this functional gradient descent approach, we can define a greedy approximation of Eq (4.3) by using a regression loss ℓ_r on the residuals computed in Eq (4.4):

$$h_t = argmin_h \sum_{i=1}^{M} \ell_r(h(x_i), r_t^i). \tag{4.5}$$

As stated above, when a weak learner h_t is updated, we need: (i) to update the weights α_t^p associated to this learner in each representation p and (ii) update the representation weights α^p in the final decision as follows:

$$\alpha^p := \alpha^p - \frac{\eta}{T}\frac{\partial \ell_c(F_t(x_i), y_i)}{\partial \alpha^p}; \quad \alpha_t^p := \alpha_t^p - \eta\frac{\partial \ell_c(F_t(x_i), y_i)}{\partial \alpha_t^p}.$$

Note that we use a learning rate $\frac{\eta}{T}$ since these weights are updated T times for a single example. All the steps of our ONLB training process are summarized in Algorithm 6.

In practice, we instantiate our losses with the square loss for regression tasks and the logistic loss for classification problems as follows:

$$\ell_c(f(x_i), y_i) = log(1 + e^{-y_i F_t(x_i)}); \quad \ell_r(f(x_i), r_t^i) = (r_t^i - f(x_i))^2.$$

The choice of the logistic loss is motivated by the need to have a bounded gradient. This avoids the residuals computed during training to grow exponentially with the iterations which can happen for noisy instances, for example. The square loss is the main loss function for regression tasks and has demonstrated superior computational and theoretical properties for the online setting (Gao et al., 2013). Then, according to Eq (4.5), the weak classifiers are updated as follows:

$$h_t = argmin_h \sum_{i=1}^{M} (h(x_i) - r_t^i)^2. \tag{4.6}$$

Eq (4.6) suggests a fairly simple update of each weak learner: each weak online learning algorithm $\mathcal{A}_i$ uses a simple online gradient descent with respect to one example at each step. The equation to obtain the residuals is straightforward:

$$r_t^i = \frac{-y_i}{1 + e^{y_i F_{t-1}(x_i)}}.$$

Algorithm 6 Online Non-Linear gradient Boosting (ONLB)

INPUT: T online weak learners, a learning rate η and P latent spaces.
Initialize $h_0 = 0$
for $i = 1$ to M **do**
  Receive example x_i
  Predict $F_0(x_i) = h_0 = 0$
  for $t = 1$ to T **do**
    Reveal y_i the label of example x_i
    Compute the residual $r_t^i = -\frac{\partial \ell_c(F_{t-1}(x_i), y_i))}{\partial F_{t-1}(x)}$
    Predict $h_t(x_i)$
    $\mathcal{A}_t$ suffers loss $\ell_r(r_t^i, h_t(x_i))$ and updates the hypothesis h_t
    for $p = 1$ to $\mathcal{P}$ **do**
      $\alpha^p := \alpha^p - \frac{\eta}{N} \frac{\partial \ell_c(F_t(x_i), y_i)}{\partial \alpha^p}$; $\alpha_t^p := \alpha_t^p - \eta \frac{\partial \ell_c(F_t(x_i), y_i)}{\partial \alpha_t^p}$
    end for
  end for
end for

Finally, we used a Rectified Linear Unit (Glorot et al., 2011), activation function such that:

$$\mathcal{L}(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The weights of the latent spaces α_t^p and α^p are now updated as follows:

$$\alpha_t^p := \alpha_t^p + \eta \begin{cases} \frac{y_i \alpha^p h_t(x_i)}{1 + e^{y_i F_t(x_i)}} & \text{if } \alpha_t^p h_t(x_i) > 0, \\ 0 & \text{otherwise} \end{cases} \quad ; \quad \alpha^p := \alpha^p + \frac{\eta}{N} \frac{y_i \mathcal{L}_p\big(\sum_{i=1}^{M} \alpha_t^p h_t(x_i)\big)}{1 + e^{y_i F_t(x_i)}}.$$

At test time, our model learned with Algorithm. 6 computes its prediction for any instance x as follows:

$$F^*(x) = sign\Big(F(x)\Big) = sign\Big(\sum_{p=1}^{\mathcal{P}} \alpha^p \mathcal{L}_p\Big(\sum_{t=1}^{T} \alpha_t^p h_t(x)\Big)\Big).$$

4.4 Extension to Batch Non-Linear Gradient Boosting

While the focus of this chapter is devoted to an online setting for non-linear gradient boosting model ONLB, our approach can be easily extended to the batch setting. We propose to present this extension in this section which will be useful later for comparison purpose. In fact, the adaptation to the batch setting is rather

straightforward and Figure 4.2 remains unchanged in this version. We present the different steps of NLB in Algorithm 7.

The learning process is very similar, however, the constraint of online weak learners is no longer valid in this context. For this reason, in NLB algorithm, and generally in gradient boosting methods, the weak learners are regression trees.

The main difference between ONLB and NLB comes from the way we learn the weak learners and update our meta learner. A new weak learner h_t is learned over the whole training set with residuals r_t^i as targets. Then, we need to find its corresponding weights $\alpha_t^p \forall p \in P$. Note that adding a new weak learner influences each representation p and their parameters α^p must be updated accordingly:

$$\alpha^p = argmin_\alpha \sum_{i=1}^{M} \ell_c(\sum_{p=1}^{P} \alpha \mathcal{L}_p\big(F_{t-1}(x_i) + \alpha_t^p h_t(x_i)\big), y_i) \tag{4.7}$$

$$\alpha_t^p = argmin_\alpha \sum_{i=1}^{M} \ell_c(\sum_{p=1}^{\mathcal{P}} \alpha^p \mathcal{L}_p\big(F_{t-1}(x_i) + \alpha h_t(x_i)\big), y_i) \tag{4.8}$$

In practice, we use a Newton Raphson step to update these weight as it has already been proposed in Friedman (2001).

Algorithm 7 Non-linear boosting

INPUT: T weak learners, $\{x_i, y_i\}_{i=1}^M$
Initialize $h_0 = 0$
Initialize α_t^p and α^p
Predict $F_0(x_i) = h_0 = 0$
for $t = 1$ to T **do**
  Compute the residuals $r_t^i = -\frac{\partial \ell_t(F_{t-1}(x_i), y_i))}{\partial F_{t-1}(x)}$
  $h_t = argmin_h \sum_{i=1}^{M} (h(x_i) - r_t^i)^2$
  for $p = 1$ to P **do**
    $\alpha^p = argmin_\alpha \sum_{i=1}^{M} \ell_c(\sum_{p=1}^{\mathcal{P}} \alpha \mathcal{L}_p\big(F_{t-1}(x_i) + h_t(x_i)\big), y_i)$
    $\alpha_t^p = argmin_\alpha \sum_{i=1}^{M} \ell_c(\sum_{p=1}^{\mathcal{P}} \alpha^p \mathcal{L}_p\big(F_{t-1}(x_i) + \alpha h_t(x_i)\big), y_i)$
  end for
  Predict $F_t(x_i)$
end for

$F(x_i) = sign\Big(\sum_{p=1}^{\mathcal{P}} \alpha_p \mathcal{A}_p\Big(\sum_{i=1}^{M} \alpha_t^p h_t(x_i)\Big)\Big)$

At test time, our model learned with Algorithm. 7 predicts exactly the same way as in ONLB:

$$F^*(x) = sign\Big(F(x)\Big) = sign\Big(\sum_{p=1}^{\mathcal{P}} \alpha^p \mathcal{L}_p\Big(\sum_{t=1}^{T} \alpha_t^p h_t(x)\Big)\Big).$$

Finally, note that our models ONLB and NLB can be easily extended to the multiclass setting, see Appendix B for more details

4.5 Experiments

In this section, we provide an experimental evaluation of our non-linear online boosting methods ONLB and NLB methods with both quantitative and qualitative analysis. We first study the batch version NLB in two steps. We begin by an intuitive illustration of the principle of NLB in the imbalanced setting and then compare this batch approach to its linear counter part over different imbalanced datasets. Second, we perform a comparative study between ONLB with different state-of-the-art online boosting algorithms on public datasets. Finally, we terminate this experimental evaluation with a qualitative analysis of the representations learned by ONLB.

4.5.1 NLB Experimental Evaluation

Graphical Visualization

In this experiment, we propose to evaluate the models with two different metrics. The first one is the F_1 score which is known to be relevant especially in the class imbalance problems where one needs to emphasize on the class of interest (usually, the positive class). We remind the F_1 score to be:

$$F_1 = 2 \times \frac{p \times r}{p + r},$$

where p and r are the precision and recall respectively.

The second evaluation metric is the Average Precision (AP), a well-known measure in the learning to rank community. We explain this choice for two main reasons. 1) It offers a good intuition of the potential of a model regardless of the decision threshold learned. Indeed, in the class imbalance setting, the decision threshold is likely to be suboptimal (see Section 2.6.2). 2) In Chapter 3 we showed that AP makes more sense when the classes are highly skewed than other metrics such as the AUCROC. We remind this measure defined as follows:

$$AP = \frac{1}{P} \sum_{i=1}^{P} p(k_i), \tag{4.9}$$

where $p(k_i)$ is the precision with respect to the rank k_i of the i^{th} positive example and P the number of positive examples.

We generate an imbalanced dataset (the red class is in minority) in two dimensions with a proportion of red points, $\pi = \frac{P}{M} = 0.1$ to highlight the main differences during the training NLB and GB. The underlying concept is rather easy with a

specificity on the top left corner where examples are randomly overlapping. Two different uniform distribution gives a probability for the blue example to appear in one of the two rectangles comprising the blue points. The same goes for the rest class.

In this experiment, both algorithms are allowed to build two weak learner (or $T = 2$) where each of them is a stump (tree with only one split). We allow 10 different representation spaces ($\mathcal{P} = 10$) for NLB. Their probability boundaries (continuous scores) are used instead of the decision boundaries (binary classification) to illustrate internal decision rules. We compute these probability boundaries as $P(y = 1|x) = \frac{1}{1+e^{F(x)}}$, where $F(x)$ is the value output given by either NLB our GB. Finally the two metrics are evaluated on the training set (using a test set was not primordial in this experiment since we can observe whether the models overfit). The results of this experiment can be observed in Figure 4.3 for NLB and Figure 4.1. A red region shows a probability $P(y = 1|x) > 0.5$ while a blue region says the opposite $P(y = 1|x) < 0.5$. A white region, on the other hand, gives a probability $P(y = 1|x) \approx 0.5$.

It is worth noticing that both algorithms learn the exact same splits. However, their weighting schema is different. In fact, GB (on the right of Figure 4.1) and NLB (Figure 4.3) build their two hypotheses naturally: first splitting the dataset vertically on x_2. Then splitting horizontally on x_1. For this second split, the only solution using the linear combination of the hypotheses is to assign more weight for the examples on the left to belong to the red class. However, this gives a higher probability for the examples on the upper left area to belong to the red class. NLB, instead, finds a representation of the hypotheses learned such as to give the highest probabilities on parts where the examples are not overlapping.

At this stage of learning, GB has a $AP = 0.4476$ while NLB has $AP = 0.9088$. The best F_1 scores for both algorithm is $F_1 = 0.7012$ and $F_1 = 0.8874$ for GB and NLB respectively. Another interesting methodological remark: The next iterations for GB are going to be more specialized on misclassified examples and thus the risk of overfitting will increase. In fact, with decision stumps, GB is not able to reach NLB's performances.

Finally, we would like to point out that, in this case, the meta-learning part does not modify the boundaries created by the weak learners. Indeed, the meta-learner does not create new boundaries but rather re-weights the existing areas as to improve the performance on the given task and so does not increase the risk of overfitting.

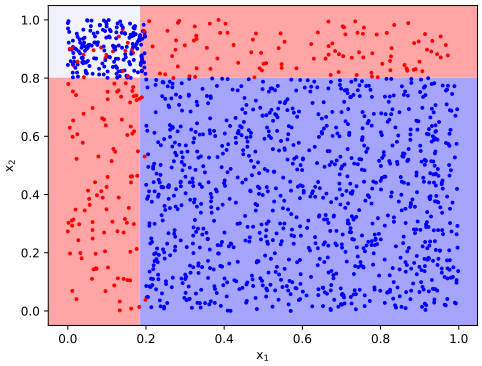

Figure 4.3: Probability boundaries of NLB on the toy dataset from Figure 4.1.

Evaluation of NLB on imbalance classification tasks

We now present an experimental study of NLB compared with GB other real datasets. We select 24 different datasets from the KEEL repository [1] that we shortly describe in Table 4.1.

Our experimental protocol generates 2/3-1/3 splits of the data to obtain the training and test sets respectively. Note that we use decision trees as our base learners. We tune the models over this 2/3 split using a 3-fold cross validation. The parameters tuned are the number of weak learners $T \in \{0, 1, ..., 100\}$, the depth of each decision tree and the learning rate. We set a maximum limit of depth 5 such as to have very weak learners. We report the averaged metrics obtained on the test sets over 30 runs in Table 4.2.

In general, NLB outperforms linear gradient boosting. Interestingly, we can see that the two metrics do not always agree on the best method. For example, for the dataset *kr-vs-k-zero vs eight*, the AP for NLB is close to perfect (0.99) and better than for GB (0.95). However, the F_1 score is much lower (0.77) for NLB than GB (0.81) on this same dataset. An explanation is that the decision threshold is not optimal regarding the F_1 score for NLB resulting in a loss in this metric. That's one of the reasons why AP is often preferable as an evaluation metric (see Chapter 3). As we discussed in Chapter 2, the metric of choice in imbalanced data is very relative to the domain (*i.e.* emphasizing more on the recall rather than the precision and vice versa) thus having very general metrics when the goal is not very well defined (emphasizing on recall rather than precision or vice versa) is important.

[1] `https://sci2s.ugr.es/keel/imbalanced.php`

Table 4.1: Properties of the datasets used in the experiments.

Dataset	#Examples	Imbalance ratio
poker-8 vs 6	1477	0.0115
abalone-20 vs 8-9-10	1916	0.0136
winequality-red-3 vs 5	691	0.0145
winequality-white-3-9 vs 5	1482	0.0169
kr-vs-k-zero vs eight	1460	0.0185
winequality-red-8 vs 6-7	855	0.0211
winequality-white-3 vs 7	900	0.0222
abalone-17 vs 7-8-9-10	2338	0.0248
kr-vs-k-three vs eleven	2935	0.0276
yeast5	1484	0.0296
winequality-white-9 vs 4	168	0.0298
yeast-1-2-8-9 vs 7	947	0.0317
poker-9 vs 7	244	0.0328
car-vgood	1728	0.0376
glass-0-1-6 vs 5	184	0.0489
zoo-3	101	0.0495
abalone9-18	731	0.0575
glass4	214	0.0607
ecoli-0-1-4-6 vs 5	280	0.0714
vowel0	988	0.0911
yeast-0-5-6-7-9 vs 4	528	0.0966
ecoli-0-1 vs 2-3-5	244	0.0984
yeast-0-3-5-9 vs 7-8	506	0.0988
yeast-2 vs 4	514	0.0992

Finally, we report in Table 4.3 the average number of weak learners and the average number of splits built by the trees to which we refer as the model complexity. We see that, on average, GB builds more complex base learners and needs almost twice as many weak learners as NLB. Also note that the model complexity depends mainly on the hyper parameter of the tree depth and that, as the depth increases linearly, the model complexity grows exponentially.

We give in Figure 4.5 the performances F_1 and AP as we add more weak learners. GB not only converges slower toward its final state but it also has an optimal solution which is less efficient than NLB. With only 15 weak learners, NLB achieves the same results as GB with 100 weak learners.

Table 4.2: The Average Precision (AP) and the F1 score (F1) reported for NLB and GB. We indicate in bold font the best method with respect to each dataset and each evaluation measure.

Dataset	NLB(AP)	GB(AP)	NLB(F1)	GB(F1)
poker-8 vs 6	$\mathbf{29.3 \pm 19.8}$	25.8 ± 31.3	$\mathbf{28.9 \pm 24.4}$	9.8 ± 19.8
abalone-20 vs 8-9-10	$\mathbf{27.9 \pm 11.7}$	20.1 ± 18.9	$\mathbf{20.2 \pm 15.7}$	19.3 ± 20.0
winequality-red-3 vs 5	8.7 ± 6.0	$\mathbf{11.1 \pm 12.3}$	$\mathbf{7.2 \pm 14.0}$	2.8 ± 7.9
winequality-white-3-9 vs 5	$\mathbf{23.8 \pm 12.6}$	14.8 ± 12.9	$\mathbf{25.8 \pm 16.9}$	14.9 ± 16.3
kr-vs-k-zero vs eight	$\mathbf{99.0 \pm 1.5}$	95.2 ± 7.0	77.1 ± 7.3	$\mathbf{81.5 \pm 16.4}$
winequality-red-8 vs 6-7	$\mathbf{13.1 \pm 8.1}$	6.8 ± 3.9	$\mathbf{12.8 \pm 13.2}$	4.3 ± 8.4
winequality-white-3 vs 7	$\mathbf{41.5 \pm 9.5}$	37.7 ± 19.2	$\mathbf{36.2 \pm 15.0}$	32.7 ± 16.5
abalone-17 vs 7-8-9-10	$\mathbf{28.7 \pm 7.9}$	21.4 ± 7.5	22.2 ± 10.2	$\mathbf{23.8 \pm 7.6}$
kr-vs-k-three vs eleven	$\mathbf{99.8 \pm 0.6}$	96.0 ± 5.1	$\mathbf{96.8 \pm 2.4}$	96.7 ± 2.8
yeast5	$\mathbf{67.2 \pm 8.2}$	62.8 ± 16.8	$\mathbf{67.6 \pm 4.6}$	62.6 ± 13.4
winequality-white-9 vs 4	$\mathbf{41.7 \pm 35.4}$	30.3 ± 34.6	$\mathbf{22.2 \pm 35.1}$	5.6 ± 15.7
yeast-1-2-8-9 vs 7	$\mathbf{29.9 \pm 12.1}$	22.2 ± 13.6	$\mathbf{25.4 \pm 14.8}$	21.2 ± 16.7
poker-9 vs 7	$\mathbf{35.1 \pm 17.1}$	25.4 ± 18.7	$\mathbf{24.1 \pm 23.0}$	15.4 ± 20.2
car-vgood	$\mathbf{99.9 \pm 0.2}$	97.3 ± 5.0	$\mathbf{96.4 \pm 4.2}$	83.2 ± 31.7
glass-0-1-6 vs 5	$\mathbf{71.2 \pm 28.9}$	65.7 ± 32.4	$\mathbf{56.3 \pm 34.4}$	36.7 ± 35.5
zoo-3	$\mathbf{35.3 \pm 29.9}$	29.4 ± 21.4	$\mathbf{32.2 \pm 30.0}$	20.4 ± 29.2
abalone9-18	$\mathbf{40.1 \pm 7.4}$	30.4 ± 9.9	$\mathbf{37.9 \pm 6.4}$	30.2 ± 11.4
glass4	$\mathbf{54.4 \pm 16.4}$	51.2 ± 22.2	46.9 ± 24.8	$\mathbf{54.0 \pm 16.1}$
ecoli-0-1-4-6 vs 5	69.9 ± 16.0	$\mathbf{74.6 \pm 18.4}$	68.9 ± 11.1	$\mathbf{69.2 \pm 11.8}$
vowel0	94.7 ± 5.2	$\mathbf{97.7 \pm 2.1}$	89.4 ± 5.8	$\mathbf{91.9 \pm 4.5}$
yeast-0-5-6-7-9 vs 4	46.8 ± 4.4	$\mathbf{55.3 \pm 12.7}$	40.3 ± 10.8	$\mathbf{52.2 \pm 12.3}$
ecoli-0-1 vs 2-3-5	$\mathbf{76.5 \pm 11.1}$	67.7 ± 11.6	$\mathbf{65.9 \pm 12.9}$	57.0 ± 8.4
yeast-0-3-5-9 vs 7-8	$\mathbf{42.1 \pm 8.3}$	36.9 ± 11.5	$\mathbf{29.4 \pm 6.9}$	29.1 ± 11.8
yeast-2 vs 4	$\mathbf{82.7 \pm 7.4}$	80.7 ± 7.4	$\mathbf{75.2 \pm 6.5}$	71.0 ± 9.6

While NLB shows a better convergence rate in terms of weak learners, it still needs an extra step to update the meta learner parameters. However, since we update our parameters sequentially and only once per weak learner and per representation, the overall update time of the meta learner is not larger than the time to train a basic neural network with one layer and T inputs (the number of weak learners).

4.5.2 ONLB Experimental Evaluation

In this section we first conduct an experiment in the online learning setting and then present an in-depth analysis of the learned representation in ONLB.

Table 4.3: Average number of weak learners and number of splits per weak learner for GB and NLB.

Model	Average #Splits	Average #Weak learners
GB	22.13 ± 7.92	67.25 ± 35.55
NLB	5.08 ± 3.83	35.42 ± 39.01

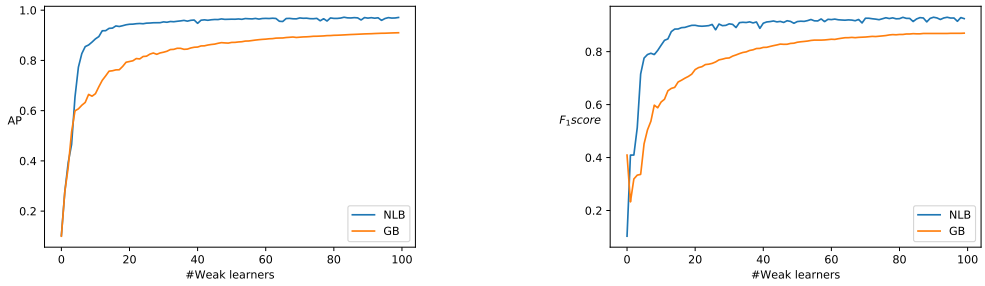

Figure 4.4: AP(on the left) and F_1 score (on the right) for NLB and GB along their iterations for MNIST dataset.

Online Learning Experiments

We use 10 public datasets from the UCI repository by considering binary classification problems where we focus on larger scale datasets than for NLB experiments that can be processed by online learning approaches (multiclass datasets were converted into binary problems as indicated in parentheses): Poker (0 vs [1,9]), MNIST ([0,4] vs [5,9]), Wine ([3,6] vs [7,9]), Abalone ([0,9] vs [10,29]), Covtype (2 vs all), Shuttle (1 vs all), Pima, Adult, HIV and w8a. A summary of these datasets is presented in Table 4.4.

Our experimental setup is defined as follows. For every dataset, we apply a 3-fold cross validation. For tuning the hyper-parameters, we perform in each fold a progressive validation (Blum et al., 1999) on the training set as proposed in (Beygelzimer et al., 2015b): when a new example arrives, it is first used to evaluate the model before the label is revealed to the learner for training. The progressive validation is a simple increment on the different quantity of the confusion matrix (TP, TP, FP, FN) given the predictions of the model on a given example before the label is revealed. Note that we simulate the online learning setting by giving the examples in a random order to the algorithm. We train different models in parallel with respect to their hyper-parameter values (*i.e.* the number of weak learners T, the learning rate η and γ the weak learner edge) and we select the one achieving the lowest progressive validation error. The selected model is then evaluated on the test set.

Table 4.4: Properties of the datasets used in the experiments.

	#examples	#features
Covtype	581,012	54
Poker	1,025,010	10
MNIST	70,000	718
Abalone	4,177	8
Pima	767	8
Adult	42,842	14
HIV	6,590	8
Shuttle	58,000	9
w8a	64,000	300
Wine	6,497	12

We compare our method to different online boosting algorithms from current state-of-the-art: the four algorithms online.BBM, Adaboost.OL, Adaboost.OL.W, OGB from Beygelzimer et al. (2015b,a) and streamBoost from Hu et al. (2017)[2].

For all the algorithms, we choose as a relatively weak classifier a neural network with one hidden layer and two units that we update in an online learning fashion using stochastic gradient descent. We report the classification error obtained for each algorithm in Table 4.5.

ONLB achieves competitive results with the state of the art online boosting methods and even outperforms them on most datasets. In some cases, such as for MNIST or Poker, we clearly see that, while using much more weak learners (see Figure 4.5), the other methods were not able to capture the target concept as much as ONLB did. Note that, a mandatory condition in our experiments was $T > 1$ such that the boosting takes part in the learning process but in some cases, the online boosting algorithms were not able to do better than the baseline on the test set. For example, on the Adult database, only ONLB and OGB achieved an average error lower than the base learner.

In Table 4.6, we present the average number of weak learners chosen with respect to the progressive validation process for each model. While being an online linear boosting algorithm, online.BBM achieves its performances with a significantly smaller number of weak learners compared to the other linear boosting methods. As mentioned in (Beygelzimer et al., 2015b), this algorithm is optimal in the sense that no online linear boosting algorithm can achieve the same error rate

[2]We used the implementations available in Vowpal Wabbit and reimplemented the streamBoost and OGB algorithms.

Table 4.5: Error rate reported for different online boosting algorithms.

Dataset	Base learner	ONLB	online.BBM	Adaboost.OL	Adaboost.OL.W	OGB	streamBoost
Covtype	0.2401	**0.2057**	0.2242	0.2273	0.2313	0.2264	0.2128
Poker	0.4182	**0.0497**	0.2375	0.1234	0.0953	0.3880	0.2668
MNIST	0.1105	**0.0561**	0.1029	0.1557	0.0830	0.1139	0.0655
Abalone	0.2673	0.2523	0.2831	**0.2487**	0.2531	0.2669	0.2720
Pima	0.2992	**0.2795**	0.2913	0.2952	0.2835	0.2874	0.2953
Adult	0.1523	**0.1465**	0.1530	0.1530	0.1526	0.1476	0.1586
HIV	0.1986	0.1393	**0.1273**	0.1360	0.1291	0.1540	0.1526
Shuttle	0.0211	**0.0024**	0.0173	0.0061	0.0058	0.0133	0.0050
w8a	0.0189	0.0148	0.0158	**0.0146**	0.0167	0.0178	0.0155
wine	0.1979	**0.1687**	0.1921	0.1931	0.1931	0.1743	0.1833

with fewer weak learners or examples asymptotically. That being said, our ONLB algorithm achieves, on average, better performance with twice less weak learners than online.BBM.

Finally, in Figure 4.5, we plot the convergence curves with respect to the increasing number of examples used for two datasets: MNIST and Abalone. For all algorithms, each curve corresponds to the evolution of the error rate according to the progressive validation error measured during training. We observe that ONLB still achieves the best convergence rate for both datasets. A similar behaviour has been observed for the other datasets and exhibits the nice fast convergence property of our algorithm which needs less weak learners to converge to its optimum.

Table 4.6: Average number of weak learners (N) selected by progressive validation.

Dataset	ONLB	online.BBM	Adaboost.OL	Adaboost.OL.W	OGB	streamBoost
Covtype	6	60	79	59	282	63
Poker	52	222	348	311	320	285
MNIST	14	66	147	207	431	131
Abalone	5	6	12	3	166	8
Pima	65	64	109	141	437	174
Adult	13	6	18	17	161	119
HIV	6	6	94	188	32	16
Shuttle	30	43	243	108	121	159
w8a	4	7	54	42	132	40
wine	5	8	112	91	97	118
Average	20	49	121	116	218	111

Analysis of the learned multi-latent representations

In this section, we present two different qualitative analyses on the latent representations learned by our algorithm. First, we show that given a sufficiently large

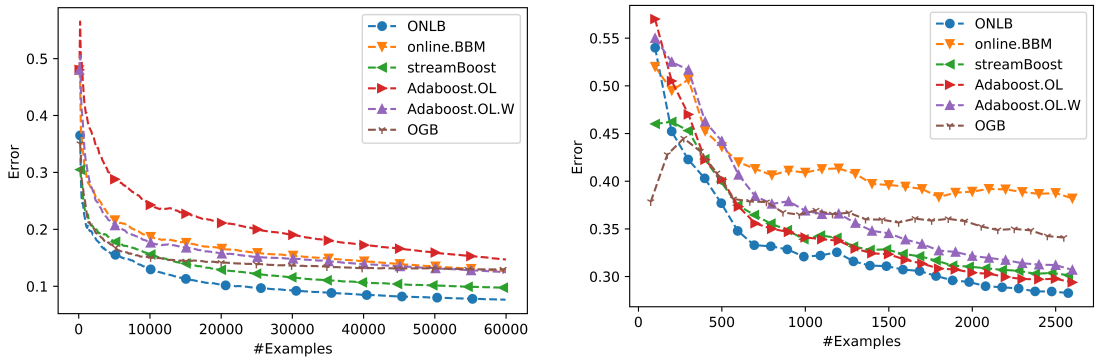

Figure 4.5: Progressive validation error with respect to the learning examples for MNIST on the left and Abalone on the right.

number of weak base learners, the representations obtained tend to be rather uncorrelated. This provides an evidence that ONLB can generate some diversity. Then, we show that these representations contribute in a comparable way to the final decision.

For our study, we use the following setup. We consider a model with 100 representations (*i.e.* $\mathcal{P} = 100$). We use two base learners: a relatively weak neural network with one hidden layer composed of 2 units (2-NN) and a stronger learner consisting of a neural network with 500 units in its unique hidden layer (500-NN). All representation weights are initialized following a uniform distribution such that the different representations are highly uncorrelated. We consider the MNIST dataset used above for learning.

Our first analysis aims at showing that the learned representations tend to be uncorrelated when using a very weak learner. For this purpose, we compute a correlation matrix C between all the representations such that $C_{nm} = \frac{cov_{nm}}{\sqrt{cov_{nn} * cov_{mm}}}$ measures the correlation between the latent representations n and m, cov is the covariance matrix computed with respect to the input weights $\{\alpha_t^m\}_{i=1}^N$ and $\{\alpha_t^n\}_{i=1}^N$ of these representations.

We show, in Figure 4.6, the C matrix for the latent space representations obtained after convergence with the 2-NN base learners. We can see that most of the representations tend to be uncorrelated or weakly correlated. In contrast, Figure 4.7 presents the C matrix using the 500-NN base learner. We see here that most of the representations are highly correlated. This experiment shows that by using sufficiently weak base learners, we are able to learn diverse and uncorrelated representations.

In our second analysis, we want to confirm that the uncorrelated latent repre-

sentations are informative enough to contribute in a comparable way to the final strong model. We propose to compute, for each representation p, a relative importance coefficient Ω_p by taking the absolute values of the predictions of p right before they are merged together with the other representation outputs to form the final prediction. We average this coefficient over $\{x_i\}_{t=1}^{K}$ examples taken from a validation set independently from the learning sample as follows:

$$\Omega_p = \frac{1}{K} \sum_{i=1}^{K} |\alpha^p \mathcal{L}_p\big(\sum_{i=1}^{M} \alpha_t^p h_t(x_i)\big)|. \tag{4.10}$$

We expect for important representations a high Ω_p (*i.e.* having a high impact in the final decision) and a low Ω_p for irrelevant ones (*i.e.* having low impact in the final decision).

We consider then the models learned with the 2-NN and 500-NN base learners as previously. For each model, we plot the importance coefficient Ω_p (y-axis) against the average correlation of each representation (x-axis) that we define as $\widehat{C}_p = \frac{1}{\mathcal{P}} \sum_{i=1}^{\mathcal{P}} C_{pi}$. This illustrates the importance of each representation in the final decision with respect to their correlation level.

Figure 4.8 gives the plot for the model using the 2-NN base learner. We see here that all the representations are involved in the final decision and that their relative importance coefficients are rather comparable.

This is in opposition to the plot of Figure 4.9 that provides the results for the model using the 500-NN base learners. First, we see that many representations are not used in the final decision and these correspond to the ones that are uncorrelated. In fact, representations involved in the final decision are the ones that are all highly correlated with an average correlation coefficient around 0.75. Clearly, since these representations have a high correlation level, actually only one representation is really useful at the end. But note that this representation can in fact be learned by a standard linear gradient boosting.

From this experiment, we see that complex models are hard to diversify in online boosting. Moreover, tuning their hyperparameters is harder making the probability of overfitting higher and they require a significant larger amount of training time which makes such complex models useless for online boosting.

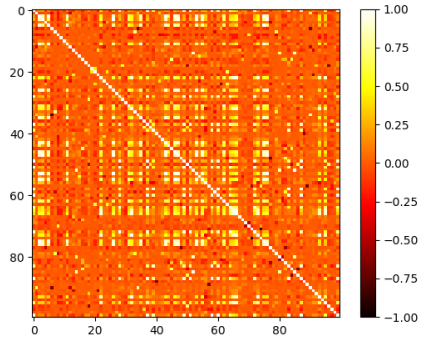

Figure 4.6: Correlation matrix of the representations with 2-NN learners.

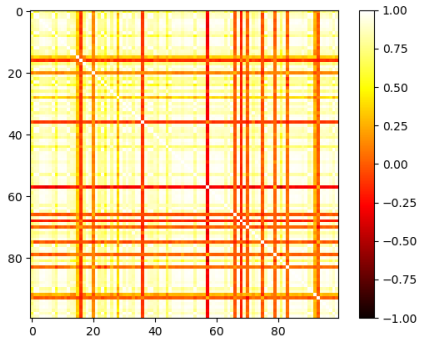

Figure 4.7: Correlation matrix of the representations with the 500-NN learners.

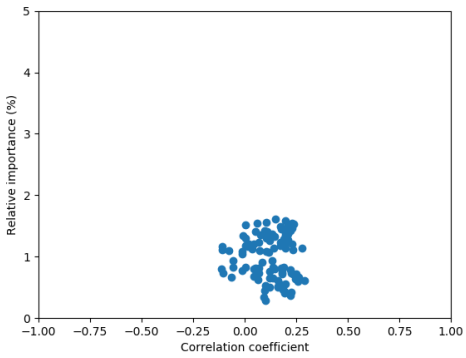

Figure 4.8: Importance of each latent representation with the 2-NN learners.

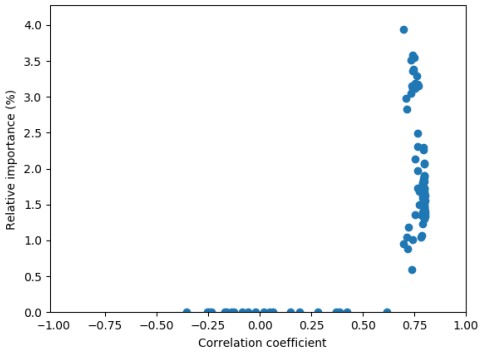

Figure 4.9: Importance of each latent representation with the 500-NN learners.

4.6 Conclusion

In this chapter, we presented a new Online Non-Linear Boosting algorithm and with its extension to the batch setting. In this method, we combine different representations of the same set of weak classifiers to produce a non-linearly boosted model in order to learn the idiosyncrasies of the target concept. Our experimental results showed that non-linear gradient boosting allows us to learn better models than classic linear gradient boosting and also exhibits a general improvement over current state of the art online boosting methods.

Additionally, the non-linear architecture of the model allows the method to use less weak learners and to obtain faster convergence in terms of examples. Our

approach has also the interesting property to produce efficiently diverse latent spaces contributing actively to the model predictions. This property makes our model adaptive by giving more importance to the best current representations. In the online learning setting, a very important point is to be able to extract as much information as possible from the examples when we receive them without overfitting.

While in our experimental online setting, we have used neural network-based weak classifiers for convenience, a first perspective is to evaluate the behaviour of our approach with other types of online weak classifiers such as non-differentiable ones (*e.g.* decision trees) allowed by our framework. In another direction, studying the impact of delayed feedback (i.e. labels arriving only after some time delay) and possible adaptation of transfer learning and continuous learning in the online setting are also particularly promising in the context of machine learning production systems such as fraud detection applications. Finally, if we take the online learning apart, we can imagine more advanced techniques such as building new architecture (Cortes et al., 2017) with this method. In this context, a general comparison with classic Neural Networks/Deep learning algorithms for example can also be an interesting future work to position our gradient boosting framework with respect to other general state of the art models.

Conclusion and Perspectives

In this manuscript, we presented two main problems that arise when using supervised machine learning for extremely imbalanced data flows such as in the task of credit card fraud detection. Before presenting the perspectives of this work, we can draw two general conclusions: one with respect to the scientific contributions of the document and another one on the impact of this thesis on the company.

Fraudulent transactions are, by nature, much less than genuine transactions. The class distribution is therefore highly skewed toward the majority class. This brings many difficulties for training machine learning algorithms that have been largely studied in the literature. However, we observe two main flaws.

1. The literature relates to relatively small datasets to validate the methods.
2. Evaluation metrics are often chosen by default which can greatly influence the conclusion.

General Discussion on the Contributions

In this thesis, we study different machine learning methods and compare their performance on real life data brought by Worldline company. It turns out that ensemble methods show a clear superiority in an extreme imbalanced context. Indeed, our experiments highlighted that Random Forest and Gradient Boosting are the most promising methods.

In Chapter 2, we carry experiments on the real-life fraud detection data using three of the most used metrics to assess the model performance on imbalanced datasets, namely AUCROC, AP and the F_1 score. We criticize the use of AUCROC to assess the model performance in case of class imbalance especially for applications that need a descent precision. We stress out the need to carefully analyze the metric of choice such as to select the most appropriate for the problem at hand which is often neglected in the literature.

In this continuum, imbalance learning methods such as sampling should also be used with caution. In Chapter 2 we presented how decision threshold dependent

metrics may be biased by a sub-optimal decision threshold. In this context, sampling methods appears to be much less efficient than simply calibrating the output probability or tuning the decision threshold on a validation set. In this sense, we recommend to use metrics independent of this decision threshold if the application allows it. In particular, ensemble methods such as gradient boosting studied in this thesis do not benefit in general from sampling methods that rather tend to worsen the results. From our analysis, we face to the problem that imbalance learning methods did not generally seem as effective as reported in the literature in the context of extreme imbalance settings which has justified the contributions of this thesis.

In the light of these findings, we proposed a first contribution to optimize the average precision, which is among the most appropriate metrics in our industrial context, for supervised anomaly detection problems in a stochastic gradient boosting algorithm. This contribution rather links the fraud detection task to the learning to rank domain. In other words, we propose to focus on the top ranked example instead of a pure classification problem. This approach also agrees with the fraud detection system in production where experts analyze the most probable fraudulent transactions before taking any action. In this contribution, we derive a smooth surrogate of the average precision and use it as a loss function.

Our second contribution addresses a negative aspect of boosting encountered in the first contribution. The classic combination of models used in gradient boosting is linear which tend to average the performance of these models instead of taking advantage of their idiosyncrasies. We propose a non-linear version of the gradient boosting algorithm. We apply this new method in the online setting such as to be able to deal with large scale uninterrupted flows of data.

Apart from these two contributions, in this manuscript we tried to follow a general guideline aiming at proposing a research driven by the need of developing novel contributions able to solve real-life data science problems which takes in particular the form for us on large imbalanced flows of data.

A Review on the Impact for the Company

Apart from the methodological scientific contributions presented above, this thesis was also the core of important new contributions for the company that can be summarized in four main aspects.

Helping to push machine learning in production such at to improve the already existing fraud detection system. This thesis is also part of a large project

where the goal is mainly to prove the effectiveness of AI methods and that its implementation into a production system is possible. In this sense, throughout this thesis, there have been many discussions with the production team to understand their needs and find solutions implying machine learning approaches. For companies that have always been relying on human expertise, such transition is not easy. At the end of this thesis, the system embedding many machine learning solutions and developed in the R&D team at Worldline has been shared with the production team which is a major step forward. In the near future, it should be integrated in the production pipeline.

The choice of a reference metric for fraud detection that followed from a large study done on imbalance learning and learning to rank. From a general perspective, the average precision (AP) has shown to have the main advantages for the credit card fraud application and is today used on a daily basis to assess the performance of the running models.

The choice of the learning algorithm of reference for the specific task of credit card fraud detection has converged to the boosting algorithm. This followed significantly higher performance in terms of both training time and predictive power on the credit card fraud detection dataset. It also is a much lighter final model than previously used learning algorithms (*e.g.* Random Forest).

Incorporation of online learning mechanisms in the pre-production system where a single model learns continuously on arriving data and predicts on the future. This in fact, demands a lot of modification on the work flow already implemented for the fraud detection task. The implementation of such system also raised a lot of different concerns such as how we assess the model's performance in real-time or how and when we update the models.

Patent on the credit card fraud detection based on the optimization of the average precision. Indeed, our view of the fraud detection application based on the learning to rank domain fits the fraud detection system where experts are given a short list of potential frauds. This led us to a write a patent that was reviewed and accepted.

Perspectives of this Thesis

This manuscript naturally leads to many different open questions induced by the contributions. First, we find necessary the thorough review of machine learning

metrics before using them in an application case. In this sense, a large study of known metrics for different use cases could be of great value for the machine learning community.

We found that the average precision is one of the best metrics for our general case, however, it has the main drawback of being highly dependent on the imbalance ratio. Further work to adapt this metric such as to make it invariant on the imbalance could be of crucial importance for many applications (e.g. monitoring through time where changes in the class distribution naturally occur).

In this manuscript we study metrics at the transaction level. It turns out that, we can compute all the metrics studied in this manuscript at the card level quite easily. However, machine learning algorithms are learning at the transaction level, therefore, using these metrics as objective functions is not straightforward and deserve further studies.

The application of fraud detection has an a delayed feedback that was not deeply studied in this thesis. However, the pace at which we receive the labels can impact the performance of our models. For this reason, online learning with delayed feedback should be further studied to understand the real impacts. The use of lifelong learning approaches also represents an appealing perspective in order to adapt the models continuously to the evolution of fraudster strategies. Indeed, some concepts may appear once per year which would be very hard for a standard online learning method to "remember" while still learning over new concepts.

As online learning gains more interest for applications such as the one studied in this manuscript, we believe that online extreme class imbalance learning would be a typical research interest to follow on our contributions. Indeed, such problems have many open questions such as finding a fair evaluation metrics that works in this context.

Finally, all contributions and discoveries in this manuscript were driven by a private dataset that comprises specific settings that are not common in the public domain. We believe that it could be of great value for the machine learning community to test their algorithms on such datasets. While sharing this data should not be done on a whim, we believe that it would have a great impact on the scientific world and therefore means should be put in place to make this happen.

List of Publications

Publications in International Conferences

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 20–35. Springer, 2017b

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Online non-linear gradient boosting in multi-latent spaces. In *International Symposium on Intelligent Data Analysis*, pages 99–110. Springer, 2018a

Publication in National Conference

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Optimisation du top rank avec le gradient boosting pour la détection d'anomalies. In *Conférence francophone sur l'Apprentissage Automatique (CAp-17)*, 2017a

Patent

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Détection par apprentissage automatique d'anomalies dans un ensemble de transactions bancaires par optimisation de la précision moyenne, Sep 2018b

International Workshop

Jordan Frery, Amaury Habrard, Marc Sebban, and Liyun He-Guelton. Non-linear gradient boosting for class-imbalance learning. In *Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 38–51, 2018c

Appendices

Appendix A

AP and F_β Score Correlation

Throughout this manuscript, we emphasize that the average precision (AP) is a better metric for evaluating the model's performance than the F_1 score or more generally the F_β score for the main reason that AP isn't biased by a sub-optimal decision threshold. It turns out that, if the decision threshold is well set, both metrics are highly correlated. This is what we want to show in this appendix.

As we presented, the average precision and the F_β score are both closely related to the precision and recall. For AP, the precision is computed at each recall level and then averaged while the F_β score corresponds to the weighted harmonic mean between precision recall at one recall level. In fact, β represents implicitly the different decision threshold.

$$AP = \frac{1}{P}\sum_{i=1}^{P} precision@k_i$$

$$F_\beta = (1+\beta^2)\frac{(1+\beta^2) \times precision \times recall}{\beta^2 precision + recall}$$

In order to compare these measures, we simulate different scores and data distribution. We set up four different types of distributions described in Table A.1.

Table A.1: Different distribution used for the simulation. N indicates a normal distribution and Beta, a beta distribution.

Positives	Negatives
N(0,1)	N(0,1)
Beta(4,1)	Beta(1,1)
Beta(1,1)	Beta(1,4)
N(3,1)	N(0,1)

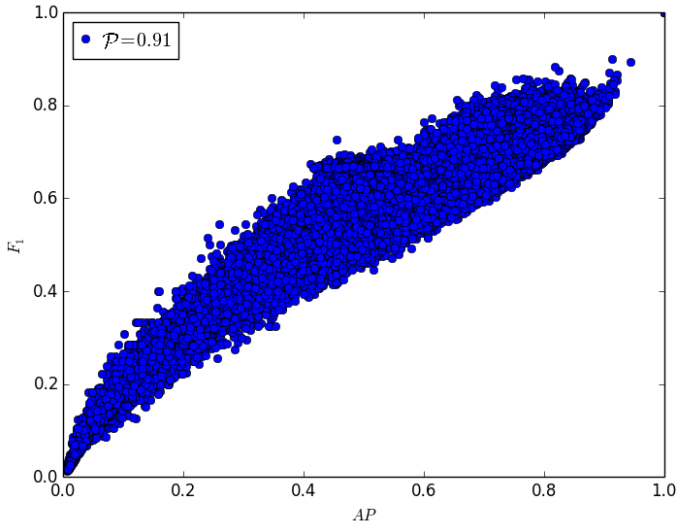

Figure A.1: Every blue dot represents a simulation. The score of AP and the best F_1 score achievable is reported for each simulation on the x-axis and the y-axis respectively.

In order to compare AP and F_1, we generate $10,000$ different datasets. For each dataset, we randomly pick one of the four distributions and generate randomly M examples with $M \in [100, 10000]$ with a positive ratio $\frac{P}{M} \in [0.01, 0.5]$ (also chosen randomly).

For each simulation, we report the AP and the best F_1 score achievable in Figure A.1. To find the latter for a given simulation, we compute the F_1 score for every decision threshold. The highest F_1 score is kept and reported in Figure A.1
.
We find that both measures are closely correlated with the Pearson correlation coefficient $\mathcal{P} = 0.91$. From this result, one can assume that optimizing AP also optimizes the best F_1 score achievable for the model in question. In fact we find similar results for different $\beta = \{0.5, 2\}$ in F_β score (see Figure A.2 and Figure A.3 respectively).
As expected, both measures are closely correlated with a Pearson correlation coefficient $\rho = 0.91$. From this result, we conclude that optimizing AP also optimizes F_β score achievable for any β with the considered model.

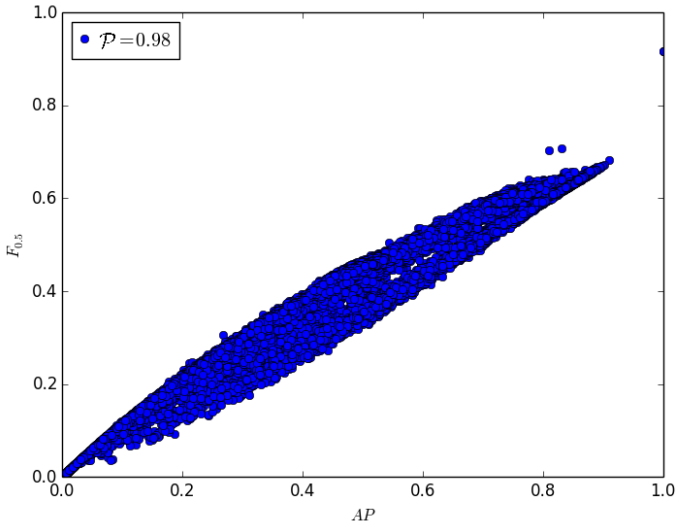

Figure A.2: Every blue dot represents a simulation. The score of AP and the best $F_{0.5}$ score achievable is reported for each simulation on the x-axis and the y-axis respectively.

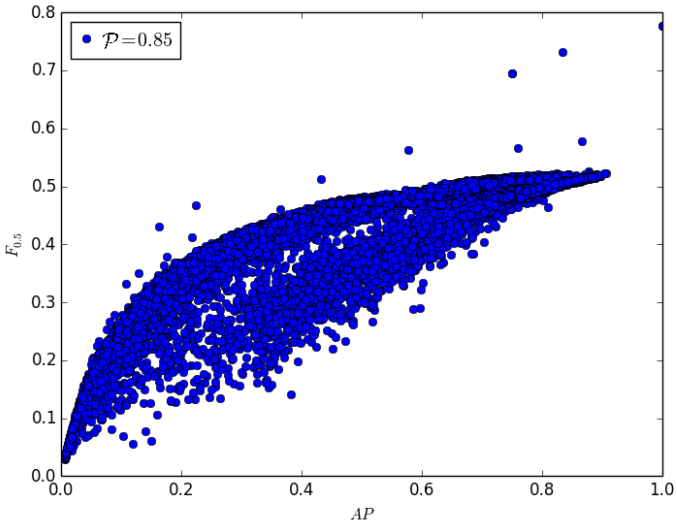

Figure A.3: Every blue dot represents a simulation. The score of AP and the best F_2 score achievable is reported for each simulation on the x-axis and the y-axis respectively.

Appendix B

ONLB in the Multi-Class Setting

An interesting extension of ONLB is its adaptation to the multiclass setting. In fact this is very straightforward since ONLB relies on a two layers neural network. In a standard neural network, multi-class problems are often solved using multiple output neurons (Bentz and Merunka, 2000; Schmidhuber et al., 2012; Ding and Dubchak, 2001). For a multiclass problem with J classes, we use J output neurons that form a vector of J dimensions representing the classes. This vector is passed through a softmax function defined as follows:

$$P(y = j|x) = \frac{e^{F^j(x)}}{\sum_{j=1}^{J} e^{F^j(x)}}$$

where $F^j(x)$ is the output of the j^{th} neuron predicting a score for sample x to belong to class j. The final modification is made on the loss function. In Chapter 1, we presented different loss functions that address the binary setting problem. Here we use a more general loss function that can be applied to any number of classes known as the multinomial logistic loss (or the cross entropy) (Böhning, 1992). First, we define our label $y_i = j$ as a one-hot vector of dimension J equal to 1 for the correct class j and 0 elsewhere. Now the loss function is defined as follows:

$$\ell_{mlog} = -\sum_{i=1}^{M} y_i \log\big(P(y_i = j|x_i)\big)$$

Then we can simply compute the derivative of ℓ_{mlog} to update the weak learners and their weights.

Bibliography

Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68:90–113, 2016. 47

Steven Abney, Robert E Schapire, and Yoram Singer. Boosting applied to tagging and pp attachment. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999. 16

Charu C Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015. 44

Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016. 44

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50. Springer, 2004. 51

Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3): 626–688, 2015. 44

Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. 36

Anup Badhe. Click fraud detection in mobile ads served in programmatic inventory. *Neural Networks & Machine Learning*, 1(1):1–1, 2017. 47

Alejandro Correa Bahnsen, Djamila Aouada, Aleksandar Stojanovic, and Björn Ottersten. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51:134–142, 2016. 50, 52

Carlos J Becker, Christos M Christoudias, and Pascal Fua. Non-linear domain adaptation with boosting. In *Advances in Neural Information Processing Systems*, pages 485–493, 2013. 93

Yves Bentz and Dwight Merunka. Neural networks and the multinomial logit for brand choice modelling: a hybrid approach. *Journal of Forecasting*, 19(3): 177–200, 2000. 125

Alina Beygelzimer, Elad Hazan, Satyen Kale, and Haipeng Luo. Online gradient boosting. In *Advances in Neural Information Processing Systems*, pages 2458–2466, 2015a. 92, 95, 106

Alina Beygelzimer, Satyen Kale, and Haipeng Luo. Optimal and adaptive algorithms for online boosting. In *ICML*, 2015b. 92, 94, 105, 106

Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *COLT*, pages 203–208. ACM, 1999. 105

Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the institute of Statistical Mathematics*, 44(1):197–200, 1992. 125

Richard J Bolton and David J Hand. Statistical fraud detection: A review. *Statistical science*, pages 235–249, 2002. 47

Francesco Bonchi, Fosca Giannotti, Gianni Mainetto, and Dino Pedreschi. A classification-based methodology for planning audit strategies in fraud detection. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 175–184. ACM, 1999. 47

Jan Brabec and Lukas Machlica. Bad practices in evaluation methodology relevant to class-imbalanced problems. *Critiquing and Correcting Trends in Machine Learning NeurIPS 2018 Workshop*, 2018. 69

Leo Breiman. Classification and regression trees. 1984. 13, 35

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 13

Leo Breiman. Arcing the edge. Technical report, Technical Report 486, Statistics Department, University of California at Berkeley, 1997. 19, 23

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 14, 25, 74

Leo Breiman and Jerome H Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American statistical Association*, 80(391):580–598, 1985. 21

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005. 75

Christopher J. Burges, Robert Ragno, and Quoc V. Le. Learning to rank with nonsmooth cost functions. In P. B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 193–200. 2007. URL `http://papers.nips.cc/paper/2971-learning-to-rank-with-nonsmooth-cost-functions.pdf`. 75, 78

Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581, 2010. 75, 77, 78, 79

Baoping Cai, Lei Huang, and Min Xie. Bayesian networks in fault diagnosis. *IEEE Transactions on Industrial Informatics*, 13(5):2227–2240, 2017. 46

James F Carney. Check fraud detection techniques using encrypted payee information, January 30 2001. US Patent 6,181,814. 47

Xavier Carreras and Lluis Marquez. Boosting trees for anti-spam email filtering. *arXiv preprint cs/0109015*, 2001. 16

Xavier Carreras, Lluís Màrquez, and Lluís Padró. A simple named entity extractor using adaboost. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 152–155. Association for Computational Linguistics, 2003. 16

Philip K Chan, Wei Fan, Andreas L Prodromidis, and Salvatore J Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications*, 14(6):67–74, 1999. 51

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009. 44

Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. pages 1–24, 2011. 74, 78

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Boosted multi-task learning. *Machine learning*, 85(1-2):149–173, 2011. 93

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *JAIR*, pages 321–357, 2002. doi: 10.1613/jair.953. URL `http://dx.doi.org/10.1613/jair.953`. 36, 74

Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *ECML PKDD*, pages 107–119, 2003. doi: 10.1007/978-3-540-39804-2_12. URL `http://dx.doi.org/10.1007/978-3-540-39804-2_12`. 31, 74

Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1): 1–6, 2004. 30

Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. *University of California, Berkeley*, 110:1–12, 2004. 39

Min Chen, Yixue Hao, Kai Hwang, Lu Wang, and Lin Wang. Disease prediction by machine learning over big data from healthcare communities. *IEEE Access*, 5:8869–8879, 2017. 46

Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. An online boosting algorithm with theoretical justifications. In *ICML*, 2012. 92, 94

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pages 785–794. ACM, 2016. 27

Davide Chicco. Ten quick tips for machine learning in computational biology. *BioData mining*, 10(1):35, 2017. 32

Michael Collins, Robert E Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285, 2002. 22

Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995. 12

Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 874–883. JMLR. org, 2017. 111

Andrea Dal Pozzolo. Adaptive machine learning for credit card fraud detection. 2015. 47

Andrea Dal Pozzolo, Olivier Caelen, Serge Waterschoot, and Gianluca Bontempi. Racing for unbalanced methods selection. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 24–31. Springer, 2013. 51, 61

Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10):4915–4928, 2014. 50, 51

Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *2015 international joint conference on Neural networks (IJCNN)*, pages 1–8. IEEE, 2015a. 50

Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. When is undersampling effective in unbalanced classification tasks? In *ECML PKDD*, pages 200–215, 2015b. doi: 10.1007/978-3-319-23528-8_13. URL `http://dx.doi.org/10.1007/978-3-319-23528-8_13`. 38, 63, 74

Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *SSCI*, pages 159–166, 2015c. 38, 63, 74

Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE transactions on neural networks and learning systems*, 29(8):3784–3797, 2018. 51, 53, 55, 61

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. pages 233–240, 2006. 69

Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000. 17

Chris HQ Ding and Inna Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001. 125

Harris Drucker and Corinna Cortes. Boosting decision trees. In *Advances in neural information processing systems*, pages 479–485, 1996. 19

Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *ICML*, pages 97–105, 1999. 16, 74

Hamid Farvaresh and Mohammad Mehdi Sepehri. A data mining framework for detecting subscription fraud in telecommunication. *Engineering Applications of Artificial Intelligence*, 24(1):182–194, 2011. 47

Peter Flach and Meelis Kull. Precision-recall-gain curves: Pr analysis done right. In *Advances in Neural Information Processing Systems*, pages 838–846, 2015. 69, 90

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Optimisation du top rank avec le gradient boosting pour la détection d'anomalies. In *Conférence francophone sur l'Apprentissage Automatique (CAp-17)*, 2017a.

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 20–35. Springer, 2017b.

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Online non-linear gradient boosting in multi-latent spaces. In *International Symposium on Intelligent Data Analysis*, pages 99–110. Springer, 2018a.

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Détection par apprentissage automatique d'anomalies dans un ensemble de transactions bancaires par optimisation de la précision moyenne, Sep 2018b.

Jordan Frery, Amaury Habrard, Marc Sebban, and Liyun He-Guelton. Non-linear gradient boosting for class-imbalance learning. In *Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 38–51, 2018c.

Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine learning*, 43(3):293–318, 2001. 17

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 12, 15, 16, 19, 92

Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Citeseer, 1996. 25

Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999. 17, 74

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003. 75, 77, 83, 87

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000. 12, 20, 22, 28

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 12, 23, 25, 75, 87, 97, 99

Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002. 24, 25, 62, 80

Jerome H Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76(376):817–823, 1981. 21

Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. Credit card fraud detection using convolutional neural networks. In *International Conference on Neural Information Processing*, pages 483–490. Springer, 2016. 50

Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. One-pass auc optimization. In *International Conference on Machine Learning*, pages 906–914, 2013. 97

Zhiwei Gao, Carlo Cecati, and Steven X Ding. A survey of fault diagnosis and fault-tolerant techniquespart i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767, 2015. 46

Nicolás García-Pedrajas, César García-Osorio, and Colin Fyfe. Nonlinear boosting projections for ensemble construction. *Journal of Machine Learning Research*, 8(Jan):1–33, 2007. 92, 95

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. 98

Helmut Grabner and Horst Bischof. On-line boosting and vision. In *CVPR*, volume 1, pages 260–267. IEEE, 2006. 16, 92, 94

Hongyu Guo and Herna L Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM Sigkdd Explorations Newsletter*, 6(1):30–39, 2004. 31

Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017. 69

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer, 2005. 36

Shizhong Han, Zibo Meng, Ahmed-Shehab Khan, and Yan Tong. Incremental boosting convolutional neural network for facial action unit recognition. In *NIPS*, pages 109–117, 2016. 93, 95

James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982. 76

Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–318, 1986. 21

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008. 30, 51

Haibo He and Yunqian Ma. *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013. 30

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE, 2008. 36

Alan Herschtal and Bhavani Raskutti. Optimising area under the roc curve using gradient descent. In *Proceedings of the twenty-first international conference on Machine learning*, page 49. ACM, 2004. 75

David J Hill and Barbara S Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25(9):1014–1022, 2010. 46

Christine Hines and Abdou Youssef. Machine learning applied to rotating check fraud detection. In *Data Intelligence and Security (ICDIS), 2018 1st International Conference on*, pages 32–35. IEEE, 2018. 47

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963. 8, 9

Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. Birdnest: Bayesian inference for ratings-fraud detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 495–503. SIAM, 2016. 47

Hanzhang Hu, Wen Sun, Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Gradient boosting on stochastic data streams. In *AISTATS*, pages 595–603, 2017. 92, 95, 106

Turker Ince, Serkan Kiranyaz, Levent Eren, Murat Askar, and Moncef Gabbouj. Real-time motor fault detection by 1-d convolutional neural networks. *IEEE Transactions on Industrial Electronics*, 63(11):7067–7075, 2016. 46

Vanita Jain. Perspective analysis of telecommunication fraud detection using data stream analytics and neural network classification based data mining. *International Journal of Information Technology*, 9(3):303–310, 2017. 47

Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26. ICST (Institute for Computer Sciences, Social-Informatics and , 2016. 46

Thorsten Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, pages 133–142, 2002. 74

Young Hun Jung, Jack Goetz, and Ambuj Tewari. Online multiclass boosting. In *Advances in Neural Information Processing Systems*, pages 920–929, 2017. 92, 95

Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre-Edouard Portier, Liyun He-Guelton, and Olivier Caelen. Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100: 234–245, 2018. 50

Yeonkook J Kim, Bok Baik, and Sungzoon Cho. Detecting financial misstatements with fraud intention using multi-class cost-sensitive learning. *Expert systems with applications*, 62:32–43, 2016. 53

Melih Kirlidog and Cuneyt Asuk. A fraud detection approach with data mining in health insurance. *Procedia-Social and Behavioral Sciences*, 62:989–994, 2012. 47

Nikolay Kotovich and Grigori Nepomniachtchi. System and method for check fraud detection using signature validation, April 10 2007. US Patent 7,201,323. 47

Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17, 2015. 45

Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016. 44

Brian Krebs. All about skimmers, June 2010. URL `https://krebsonsecurity.com/all-about-skimmers/`. 49

Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Nashville, USA, 1997. 30

Pallavi Kulkarni and Roshani Ade. Logistic regression learning model for handling concept drift with unbalanced data in credit card fraud detection system. In *Proceedings of the Second International Conference on Computer and Communication Technologies*, pages 681–689. Springer, 2016. 50, 61

C. Leistner, A. Saffari, P. Roth, and H. Bischof. On robustness of on-line boosting - a competitive study. In *3rd ICCV Workshop on On-line Computer Vision*, 2009. 95

Kevin J Leonard. Detecting credit card fraud using expert systems. *Computers & industrial engineering*, 25(1-4):103–106, 1993. 49

Nan Li, Rong Jin, and Zhi-Hua Zhou. Top rank optimization in linear time. In *NIPS*, pages 1502–1510, 2014a. 87

Weixin Li, Vijay Mahadevan, and Nuno Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):18–32, 2014b. 46

Paulo JG Lisboa, Bill Edisbury, and Alfredo Vellido. *Business applications of neural networks: the state-of-the-art of real-world applications*, volume 13. World scientific, 2000. 56

Tie-Yan Liu. *Learning to Rank for Information Retrieval.* Springer, 2011. ISBN 978-3-642-14266-6. 74

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009. 39, 40

Herman Lopata. Fraud resistant credit card system, February 3 1987. US Patent 4,641,017. 49

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 57

Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000. 27

Guillaume Metzler, Xavier Badiche, Brahim Belkasmi, Elisa Fromont, Amaury Habrard, and Marc Sebban. Tree-based cost sensitive methods for fraud detection in imbalanced data. In *International Symposium on Intelligent Data Analysis*, pages 213–224. Springer, 2018. 53

Eric WT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems*, 50(3):559–569, 2011. 47

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *ICML*, pages 625–632, 2005. 74

Nikolaos Nikolaou, Narayanan Edakunni, Meelis Kull, Peter Flach, and Gavin Brown. Cost-sensitive boosting algorithms: Do we really need them? *Machine Learning*, 104(2-3):359–384, 2016. 40, 61

Bing Niu, Yu-Dong Cai, Wen-Cong Lu, Guo-Zheng Li, and Kuo-Chen Chou. Predicting protein structural class with adaboost learner. *Protein and peptide letters*, 13(5):489–492, 2006. 16

Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier-boosting independent embeddings robustly. In *CVPR*, pages 5189–5198, 2017. 93, 95

Nikunj C Oza. Online bagging and boosting. In *Systems, man and cybernetics, 2005 IEEE international conference on*, volume 3, pages 2340–2345. IEEE, 2005. 92, 94

T Maruthi Padmaja, Narendra Dhulipalla, Raju S Bapi, and P Radha Krishna. Unbalanced data classification using extreme outlier elimination and sampling techniques for fraud detection. In *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, pages 511–516. IEEE, 2007. 51

Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing f-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, pages 2123–2131, 2014. 38

Raghavendra Patidar, Lokesh Sharma, et al. Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(32-38), 2011. 49

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. 57

Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *Acm sigkdd explorations newsletter*, 6 (1):50–59, 2004. 47, 51

Foster Provost. Machine learning from imbalanced data sets 101. 38

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 13

J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987. 13

Enislay Ramentol, Yaile Caballero, Rafael Bello, and Francisco Herrera. SMOTE-RSB *: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. *Knowl. Inf. Syst.*, pages 245–265, 2012. doi: 10.1007/s10115-011-0465-6. URL `http://dx.doi.org/10.1007/s10115-011-0465-6`. 74

Joseph Raphson. *Analysis aequationum universalis seu ad aequationes algebraicas resolvendas methodus generalis, & expedita, ex nova infinitarum serierum methodo, deducta ac demonstrata*. typis Tho. Braddyll, prostant venales apud Johannem Taylor, ad insigne Navis , 1697. 21

Vipula Rawte and G Anuradha. Fraud detection in health insurance using data mining techniques. In *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, pages 1–5. IEEE, 2015. 47

Abhimanyu Roy, Jingyi Sun, Robert Mahoney, Loreto Alonzi, Stephen Adams, and Peter Beling. Deep learning detecting fraud in credit card transactions. In *Systems and Information Engineering Design Symposium (SIEDS), 2018*, pages 129–134. IEEE, 2018. 50

Yusuf Sahin, Serol Bulkan, and Ekrem Duman. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923, 2013. 52

Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015. 69

Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2): 197–227, 1990. 15

Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999. 17, 25

Robert E Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998. 19

J Schmidhuber, U Meier, and D Ciresan. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. IEEE, 2012. 125

Martin Scholz and Ralf Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007. 93

Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Trans. Systems, Man, and Cybernetics*, (1):185–197, 2010. doi: 10.1109/TSMCA.2009.2029559. URL `http://dx.doi.org/10.1109/TSMCA.2009.2029559`. 74

Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316. IEEE, 2010. 46

Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40 (12):3358–3378, 2007. 31

Ivan Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on systems, Man, and Cybernetics*, (6):448–452, 1976. 36

Isaac Triguero, Sara del Río, Victoria López, Jaume Bacardit, José M Benítez, and Francisco Herrera. Rosefw-rf: the winner algorithm for the ecbdl14 big data competition: an extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems*, 87:69–79, 2015. 44

Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36 (10):11994–12000, 2009. 46

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142, 1984. 9

Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, 2015. 50

Véronique Van Vlasselaer, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Gotcha! network-based fraud detection for social security fraud. *Management Science*, 63(9):3090–3110, 2016. 47

VN Vapnik. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–281, 1971. 9

Stijn Viaene, Richard A Derrig, and Guido Dedene. A case study of applying boosting naive bayes to claim fraud diagnosis. *IEEE Transactions on Knowledge and Data Engineering*, 16(5):612–620, 2004. 16

Paul a Viola and Michael J Jones. Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade. *proc. NIPS*, (December):1311–1318, 2001. ISSN 1049-5258. 16

John Wallis, John Playford, Richard Davis, and Nicolas Fatio De Duillier. *A Treatise of Algebra, Both Historical and Practical*. John Playford, 1685. 21

Yibo Wang and Wei Xu. Leveraging deep learning with lda-based text analytics to detect automobile insurance fraud. *Decision Support Systems*, 105:87–95, 2018. 47

Wei Wei, Jinjiu Li, Longbing Cao, Yuming Ou, and Jiahang Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4):449–475, 2013. 44

Gary M Weiss. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19, 2004. 31

Christopher Whitrow, David J Hand, Piotr Juszczak, D Weston, and Niall M Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55, 2009. 50

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992. 14

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010. 75, 78, 87

Miao Xie, Song Han, Biming Tian, and Sazia Parvin. Anomaly detection in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 34(4):1302–1325, 2011. 46

Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR*, pages 391–398. ACM, 2007. 75

Hualong Yu, Changyin Sun, Xibei Yang, Wankou Yang, Jifeng Shen, and Yunsong Qi. Odoc-elm: Optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data. *Knowledge-Based Systems*, 92:55–70, 2016. 38

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR*, pages 271–278, 2007. 74

Evangelia I Zacharaki, Sumei Wang, Sanjeev Chawla, Dong Soo Yoo, Ronald Wolf, Elias R Melhem, and Christos Davatzikos. Classification of brain tumor type and grade using mri texture and shape in a machine learning scheme. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 62(6):1609–1618, 2009. 45

Masoumeh Zareapoor, Pourya Shamsolmoali, et al. Application of credit card fraud detection: Based on bagging ensemble classifier. *Procedia Computer Science*, 48(2015):679–685, 2015. 51, 61